Constrained Surface Parameterization Methods with Guarantees

by

Hanxiao Shen

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

September, 2022

Professor Denis Zorin

Professor Daniele Panozzo

Acknowledgements

It would be impossible for me to finish this dissertation without the help and encouragement from all my friends, colleagues, and advisors. First, I would like to express my deepest gratitude to my two great advisors, Prof. Daniele Panozzo and Prof. Denis Zorin, for leading me through the journey and providing guidance with superior expertise and patience. And to my dissertation committee members, Prof. Marcel Campen, Prof. Ken Perlin, and Prof Claudio Silva, for their time and patience. I would also like to express my sincere gratitude to my friends Zhe Wang, Xiaoyang Lu, Xiaoyang Fang, and Yuchang Chen. Without their company, my life would be too dull to endure. And I am thankful to my great collaborators Zhongshi Jiang, Leyi Zhu, Ryan Capouellez, and Xifeng Gao for valuable discussions and warmth encouragement. Also, I would like to thank all the members of the Geometric Computing Lab at NYU. It is my honor to be a part of this fantastic family. Last but not least, I want to thank my family, especially my parents, who are always there for me. I am in their debt forever for all they have sacrificed for me to pursue my dream.

Abstract

Surface parameterization for piecewise-linear surfaces is a fundamental problem in computer graphics and geometry processing. The generation of surface parameterization is a key step in numerous applications like texture mapping, remeshing, quadrangulation, inter-surface mapping, and shape-analysis. Due to its popularity, the robustness of mapping generation methods plays a major role in its applicability. In addition, depending on the specific requirements of the application at hand, various formulations of constraints are used to control or guide the parameterization. Typical examples of the constraints are point constraints, curvature constraints, and topological constraints. In many practical cases, to ensure that the input assumptions of downstream algorithms are satisfied, the constraints, such as need to be imposed exactly (as opposed, e.g., to approximation via penalties). In this work, we investigate different constraint formulations suitable for various applications and present algorithms with guarantees to generate parameterization fully satisfying these constraints. In the first part of this thesis, we develop an algorithm that solves the classical problem of mapping a disk domain with boundary constraints; in the special case of domains with convex boundary, it improves, in terms of robustness, on the classical Tutte's algorithm. Utilizing it as a building block, we design a parameterization method that supports arbitrary positional constraints. In the second part, building on recent developments in the theory of discrete uniformization, we develop a highly robust algorithm for discrete conformal maps that satisfy prescribed curvature constraints. In the third part, we provide a constructive proof for the existence of globally seamless parameterization that matches admissible

user-prescribed cone position and curvature constraints. Lastly, we generalize this to constraints on holonomy angles on a homology basis of loops, which fully capture the topology of seamless parameterizations. This method yields parameterizations that are very close to field-aligned parameterizations obtained using commonly used methods but, in contrast to these methods, guarantees the existence of solution satisfying all constraints.

Contents

List of Figures

List of Tables

1 Introduction

Surface parameterization technique is a fundamental tool in computer graphics, where most commonly, the goal is to generate piecewise-linear surface-to-plane maps. In this thesis, we are concerning triangle meshes, where the topology is encoded as simplicial complexes with degree zero, one, and two, which correspond to vertices, edges, and faces, respectively. The geometric properties of triangular meshes are determined by the location of vertices of the mesh, represented as 3D coordinates. This map has broad applicability in geometry processing, e.g., texture mapping, remeshing, inter-surface mapping, morphing, and quadrangulation. The efficiency and reliability of these methods highly hinge on the robustness and quality of the map. To reliably generate surface parameterization serving the need of different types of applications has been of significant interest for the past decades. However, there are still weak spots in existing methods where user controls are limited or no hard constraints are provided. This thesis aims to investigate different formulations of user controls and how to enforce them while maintaining common properties for parameterization like orientation preservation and quality optimization.

1.1 Outline

In this thesis, our primary focus is on how to reliably construct a surface parameterization that satisfies user-prescribed hard constraints in different forms, i.e., fixed boundary, positional constraints, curvature constraints, and topological constraints. We first provide an overview of major

aspects in the realm of surface parameterization that are our key focus.

Bijectivity and positional constraints. One of the essential properties of surface parameterization is bijectivity. In this case, it is required that the Jacobian of the linear map have a positive determinant for each element. In addition, if the boundary of the image under the map does not intersect with itself, a global bijective is achieved[Jiang et al. 2017]. This property is vital for applications like texture mapping fig. 1.2, because the map is sampled over the domain to get color values for texturing the surface. Ambiguity is created if triangles overlay each other. It is well known for 3-connected disc topology meshes, [Tutte 1963a] can map it to the interior of a convex polygon on the 2D plane with theoretical guarantees. Tutte's embedding is very popular for its simplicity and theoretical guarantees for bijectivity. One natural question is to ask how well it works in practice using standard floating points. In the first part of the thesis, we investigate this problem over a large dataset [Zhou and Jacobson 2016] and discovered its numerical bottleneck. Positional constraints are one of the most straightforward type of user control for additional guidance for the parameterization, where the location in the parametric domain for a subset of vertices are predetermined. These constraints are great supports in texture mapping. For example, feature points for matching certain parts (e.g., the eyes of the head model and the eye of the tiger, fig. 1.2). Location control could also be used to improve the packing efficiency of texture alas inside squared images. In addition to a sparse set of control points, they can also be grouped together as poly-lines to support feature curves, as shown in fig. 1.4, or be further extended to restrict the whole boundary of the input mesh as a generally shaped polygon. Instead of constraining the boundary as a convex polygon, utilizing our framework and adapting the [Kraevoy et al. 2003], we are able to generate the map for more general shaped polygons (self-overlapping [Weber and Zorin 2014]) while providing positional constraints for users.

Curvature Constraints. In a surface parameterization problem, the variables are typically set to be the 2D locations of the vertices. For example, in [Tutte 1963a] the problem is reduced to

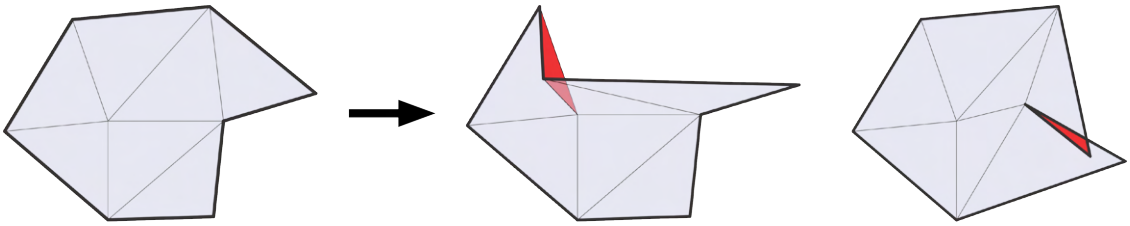


Figure 1.1: Map introduces flipped triangle, i.e. the orientation of the red triangle is not positive, or causes overlapping while maintain locally injective.

Figure 1.2: Positional constraints enables users to control the location of a set of vertices (control points, blue) on the plane.

solve a system of linear equations where the coordinates for each vertex are directly the solution. This formulation is both clean and straightforward. However this framework is limited to disk topology. Meshes with arbitrary topology are usually handled by first cut along an appropriate set of seam edges, namely a cut-graph, to convert it to one or several patches and then parameterize individually, fig. 1.3. The discontinuity across the cut edges are especially hard to handle. Researchers have been looking for alternatives for this problem for the past few decades. The notion of cone singularities was then proved to be a perfect candidate for this task [Kharevych et al. 2005; Springborn et al. 2008]. The core idea is that instead of solving for 2D locations of vertices, the variables per-vertex scale factors that are used for scaling of original edge lengths, which then defines a metric over the mesh. The primary goal is to find a flat cone metric where all the Gaussian curvature are concentrated at a few locations called cones under this metric. After that a cut can be generated to unfold the metric onto the plane and produce a parameterization. Notice this time the cuts are introduced after the flat cone metric is produced, thus the edge lengths across the cuts are guaranteed to be the same. In the second part of the thesis, we provide

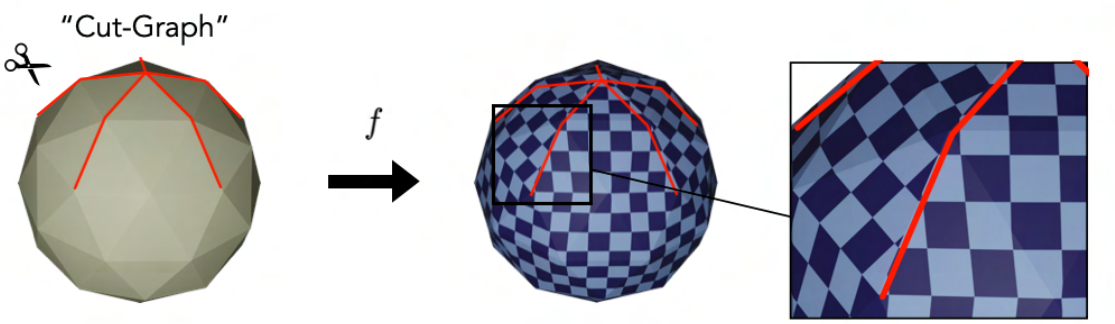


Figure 1.3: A sphere mesh is cut open along the red edges then parameterize. Notice there are discontinuity across the edges, e.g. the parametric lines do not match.

a robust framework for computing such a metric efficiently and reliably, with the guarantee for respecting user-prescribed curvature constraints.

Holonomy Constraints. When a cut-graph is applied to a surface, the parameterization results unavoidably have discontinuity across the cut edges. The term seamless parameterization is introduced for a special class of such maps where the edges on two sides of the cuts in the parametric domain are related by a rigid transform, and furthermore, the rotation in the transform is multiples of 90 degrees. For seamless parameterization, the topological control over so called holonomy is crucial. This quantity measures the angle between the initial pose and final pose of a given vector if parallel transported around a closed curve. As pointed out by [Myles and Zorin 2012], the topology of a given seamless parameterization is fully captured by holonomy defined along closed loops on the surfaces. To be more specific, homology basis loops and loops around singularities (where the total angle sum around vertices in the parametric domain is not 2π) of the parameterization. These singularities will match with the extraordinary vertices in a quadrangulation, which is a typical downstream application for seamless parameterizations with additional quantization steps ([Ebke et al. 2013]). The location, as well as the curvature of these singularities, largely determines the structure of the quadrangulation. In the third part of the thesis, we provide a constructive proof to show the existence of seamless parameterization

given a set of admissible cones that respects the Gauss-Bonnet Theorem. This set of constraints is local because they only involve small loops around the singularity, s.t., there is only one single singularity surrounded by these loops. When taking the holonomy around global homology basis loops into account, the topology of the seamless parameterization is then fully determined. In the final part of the thesis, we show a method for constructing a seamless locally injective parameterization that fully matches user prescribed holonomy signature both in the local and global sense, where holonomy over homology basis loops are taken into account.

Material presented in this dissertation have previously appeared in the following resources:

- Hanxiao Shen, Zhongshi Jiang, Denis Zorin, and Daniele Panozzo. 2019. Progressive embedding. ACM Trans. Graph. 38, 4, Article 32 (August 2019), 13 pages.
  https://doi.org/10.1145/3306346.3323012

- Marcel Campen, Ryan Capouellez, Hanxiao Shen, Leyi Zhu, Daniele Panozzo, and Denis Zorin. 2021. Efficient and robust discrete conformal equivalence with boundary. ACM Trans. Graph. 40, 6, Article 261 (December 2021), 16 pages.
  https://doi.org/10.1145/3478513.3480557

- Marcel Campen, Hanxiao Shen, Jiaran Zhou, and Denis Zorin. 2019. Seamless Parametrization with Arbitrary Cones for Arbitrary Genus. ACM Trans. Graph. 39, 1, Article 2 (February 2020), 19 pages.
  https://doi.org/10.1145/3360511

- Hanxiao Shen, Leyi Zhu, Ryan Capouellez, Marcel Campen, Daniele Panozzo, and Denis Zorin. 2022. Which Cross Fields can be Quadrangulated? Global Parameterization from Prescribed Holonomy Signatures.
  https://doi.org/10.1145/3528223.3530187.

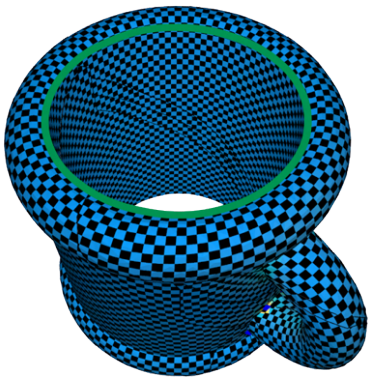


Figure 1.4: Parameterization example where the parametric lines are aligned with green feature curve.

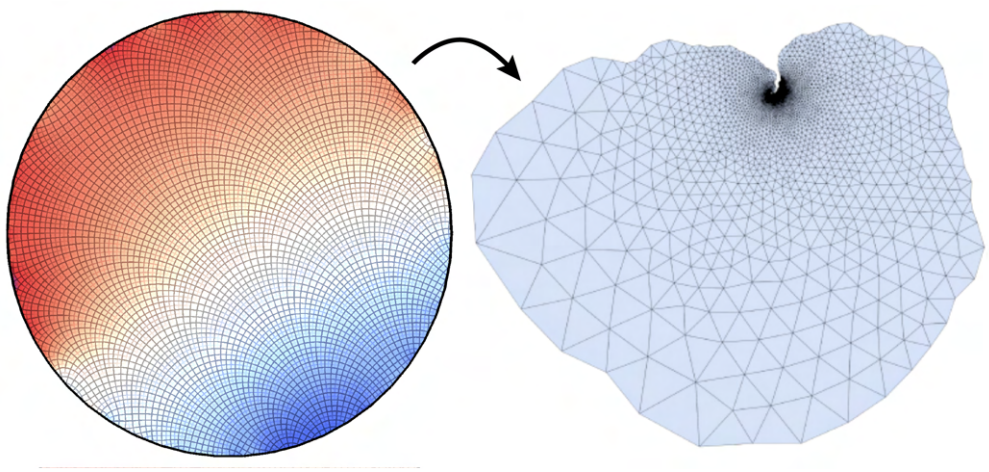


Figure 1.5: Conformal mapping generated with randomly prescribed geodesic curvature along the boundary, color visualized with per-vertex scale factor which induces conformal metric.

2 Related work

2.1 Bijective Maps

2.1.1 Planar Embedding of Graphs and Meshes

Fary's theorem [Fáry 1948] states that any planar graph can be embedded in the plane with straight edges. Tutte [Tutte 1963a] extends this result to the case of fixed convex boundary with a spring analogue, and [Floater 1997a] established its connection to the parameterization methods in the geometry processing community, and extend Tutte's uniform weight to arbitrary positive ones. In both cases, the problem is reduced to solving a linear system of equations and the resulting embeddings' minimum area might even be negative exponential with respect to the number of vertices. There has been active effort in the graph drawing community to address these issues, by bounding the total area when drawing on integer grids, or equivalently, controlling the minimum resolution [Chambers et al. 2011] under fixed diameter. Most notably, [Schnyder 1990] shows an algorithm to embed a planar graph onto integer grids inside a triangle region, and [Chambers et al. 2011] proves an upper polynomial bound on the area while keeping a specified convex boundary shape: the proof is constructive and may (potentially) be used as a basis for a practical algorithm. In all cases, rounding problems will affect these algorithms as the size of the graph grows (Section 3.2.1).

Orbifold Tutte Embedding. Multiple extensions of Tutte's theorem to map surfaces to different co-domains have been proposed. In particular, the theorem has been extended to map surfaces to a Euclidean orbifold [Aigerman and Lipman 2015a], to a hyperbolic orbifold [Aigerman and Lipman 2016a], and to a spherical orbifold [Aigerman et al. 2017]. All three methods support hard positional constraints and ensure the generation of a bijective map between the surface and the orbifold in infinite precision arithmetic. These methods also suffer from similar numerical issue as Tutte's, and extending our algorithm to orbifold embeddings is an interesting direction for future work.

2.1.2 Progressive Meshes

The well-known progressive meshes algorithm [Hoppe 1996; Sander et al. 2001] shows how a triangle mesh can be simplified by collapsing one edge at a time, and reconstructed applying the inverse topological operations in the inverse order. This scheme has been introduced as an efficient way to store, transmit, and render large meshes, where the per-vertex properties of the removed vertex are stored together with the information required to insert them back. This work has been later applied to compute inter-surface mappings [Schreiner et al. 2004], by jointly simplifying two meshes into a common base mesh, then starting from optimizing their isometric distortion while reinserting the vertices in the base mesh.

We use the same idea to eliminate problematic regions of an existing embedding (either flipped, or with a high distortion), and then reinserting one vertex at a time, while preserving the quality of the triangulation. Differently from progressive meshes, in our case we do not have geometrical information available that could help us decide where the vertex should be inserted to obtain a valid embedding.

2.1.3 Distortion-Minimizing Mappings

In this section, we focus on the recent works closely related to generating distortion-minimizing discrete locally injective and globally bijective discrete maps, and we refer to [Floater and Hormann 2005b; Sheffer et al. 2006; Hormann et al. 2007] for a comprehensive treatment of earlier parametrization methods without these properties.

A discrete locally injective map requires that triangles maintain their orientation (i.e. they do not flip) and if the sum of (unsigned) triangle angles around each internal vertex is precisely 2π [Weber and Zorin 2014]. Three main families of methods have been proposed to deal with this challenging constraint: barrier, convexification, and hybrid algorithms.

Barrier Algorithms. Barrier algorithms require a valid initial solution, and then optimize its quality without leaving the feasible space. The key idea is to adopt quality metrics diverging to infinity when triangles become degenerate, thus inhibiting flips. Popular choices strive to preserve angles [Hormann and Greiner 2012; Degener et al. 2003] or lengths [Sander et al. 2001; Sorkine et al. 2002; Aigerman et al. 2014; Poranne and Lipman 2014; Smith and Schaefer 2015]. Alternatively, a barrier functions can be added to existing energies to enforce local injectivity [Schüller et al. 2013]. These non-linear energies are difficult to minimize, stemming a series of methods specifically targeting this problem. They include coordinate descent [Hormann and Greiner 2012; Labsik et al. 2000], parallel gradient descent [Fu et al. 2015a], Anderson Acceleration [Peng et al. 2018], as well as other quasi-newton approaches [Smith and Schaefer 2015; Kovalsky et al. 2016a; Rabinovich et al. 2017a; Shtengel et al. 2017a; Claici et al. 2017; Zhu et al. 2018a; Liu et al. 2018].

All these methods support hard-constraints if they are already satisfied in the initial map, which is the key idea used in MatchMaker [Kraevoy et al. 2003]. Our progressive embedding can be used to robustly generate the initial map, that can then be improved by any of the previous techniques (Section 3.4).

Projection Algorithms. An essential component of these methods is a convexified form of the injectivity constraints [Lipman 2012; Kovalsky et al. 2015]. While these methods naturally support hard injectivity constraints, they might fail to find a feasible solution, with no output generated. The only known way to guarantee that a feasible solution exists is to formulate the convexified constraints using a reference frame derived from a valid (although potentially very high distortion) solution.

Hybrid Algorithms. Hybrid algorithms are an interesting mix between these two approaches [Fu and Liu 2016; Poranne et al. 2017]. The initial guess is produced by separating all triangles and isometrically rotating them into the UV space. A barrier method is then used to prevent them from flipping, while trying to seal the seams. This approach might fail to seal all the seams, not producing a valid map.

Globally Bijective Maps. For simply connected domains, bijective maps are locally injective maps whose boundary does not intersect. All embeddings described in Section 2.1.1 satisfy this property. These methods have been extended to non-convex, self-overlapping polygons [Weber and Zorin 2014] and polyhedrons [Campen et al. 2016], but they still require a fixed boundary. Few methods can produce bijective maps while letting the boundary free, relying on either collision detection [Smith and Schaefer 2015] or scaffolding elements [Gotsman and Surazhsky 2001; Zhang et al. 2005; Müller et al. 2015; Jiang et al. 2017]. All free boundary methods require a starting point: our algorithm can be used to generate it, enabling these algorithms to create bijective maps with hard constraints (Section 3.3).

Hard Positional Constraints and Refinement. Matchmaker [Kraevoy et al. 2003] introduced hard positional constraints for texture mapping applications. The algorithm uses a two-step approach, first generating a valid map, and then optimizing its geometrical quality. The method is one of the few using refinement to guarantee the existence of feasible solutions. The

method has been extended by adding an intermediate warping stage to align the constraints in [Lee et al. 2008]. We show in section 3.3 how our embedding can be used withtin Matchmaker to increase its robustness, and we also show how to extend Matchmaker to support self-overlapping polygonal target domains.

Cross-Parametrization. Cross-parametrization, i.e. the computation of a map between two surfaces, is another problem that often relies on planar embeddings. [Schreiner et al. 2004] and [Kraevoy and Sheffer 2004] proposed the first provably guaranteed solutions to compute maps between surfaces, by reducing the problem to mapping both surfaces to a common subdomain by either using Tutte's embedding or a simplification approach. A similar construction that cuts open the surface into a single topological disc has been proposed in [Aigerman et al. 2014], and extended to allow even the optimization of the seams positions in [Aigerman and Lipman 2015b]. Floating point rounding errors have not been considered in any of these works, which are more prone to fail as the resolution of the mesh increase or whenever the user-provided constraints introduce a high distortion (Section 3.4).

Global Parametrization. Field-aligned parametrization methods [Bommes et al. 2009a] strive to compute a locally injective map [Bommes et al. 2013a] whose gradient is aligned with a user-provided directional field. We refer an interested reader to [Bommes et al. 2012] for a comprehensive overview of these techniques. Our embedding algorithm can be used to compute parametrizations to a target self-overlapping polygon, enabling to robustly generate these maps if a valid boundary polygon is provided (Section 3.4).

2.2 Conformal equivalent metrics

The problem of computing conformally equivalent metrics or, by implication, conformal maps of discrete surfaces, has been considered in a variety of works before. As there is no useful natural

notion of conformality in the discrete (non-smooth) setting, a range of discrete counterparts of the continuous concept of conformality have been proposed and used.

Static Triangulation. Prominent examples of works addressing the computation of conformal metrics or conformal maps on discrete surfaces, based on various definitions of discrete conformality, while considering the triangulation fixed are based on least-squares formulations [Lévy et al. 2002a; Desbrun et al. 2002], vertex scaling formulations [Springborn et al. 2008; Ben-Chen et al. 2008; Sawhney and Crane 2017; Jin et al. 2007; Soliman et al. 2018a], circle patterns [Kharevych et al. 2006a], or formulations based on holomorphic one-forms [Gu and Yau 2003].

Dynamic Triangulation. A fixed triangulation restricts the metric space that can be achieved. By adjusting the triangulation depending on the prescribed target curvature, this limitation can be remedied. Two systematic approaches have been proposed to that end, both conceptually considering a continuous metric evolution from initial state to target state. [Luo 2004] proposes to adjust the triangulation by an intrinsic edge flip whenever an edge becomes triangle inequality critical (4.1 left). Implementation variants are described in [Campen and Zorin 2017b,0; Campen et al. 2019]. Differently, [Gu et al. 2018b,0; Springborn 2019] effectively consider the case of flipping an edge when it becomes Delaunay-critical, i.e., when four vertices become co-circular (4.1 right). Surfaces with boundary in this context are addressed in [Sun et al. 2015] using a double cover approach, reducing this case to the case without boundary. A correspondence map between the original triangulation and the modified triangulation can be kept track of by means of an overlay data structure [Fisher et al. 2007].

In concurrent work, [Gillespie et al. 2021] make use of the same theoretical results we use here and describe an algorithm that conceptually is very close to our core algorithm in 4.3. Main differences of our work are (i) a number of important details in the optimization procedure as described in 4.3, (ii) special combinatorial handling of symmetry in the double surface used to

support meshes with boundary, and (iii) extensive evaluation in particular of numerical limits and numerical precision effects. In comparison, [Gillespie et al. 2021] propose a more lightweight data structure (than [Fisher et al. 2007] that we use) to keep track of the mesh overlay, and additionally consider the case of spherical parametrization.

2.3 Seamless Parameterization

Seamless surface parametrization and the related subject of quadrangulation and quad layout generation is a well-explored topic. A relatively recent survey [Bommes et al. 2013c] has references to many works in this area. We focus here on the most closely related ones.

In a wide variety of applications, surface parametrizations are required to be (locally) injective (i.e., without fold-overs) as well as to exhibit low parametric distortion [Floater and Hormann 2005a]. Due to the challenging nature of this requirement, a common strategy is to proceed in a two-step fashion: first construct an initial injective parametrization (without specific attention to distortion), then optimize it with respect to application specific distortion criteria (while preserving injectivity). Our work follows this strategy.

Constructing injective maps. Whenever a robust overall algorithm is desired, injective maps are almost always initialized using the same classical result on convex harmonic maps [Tutte 1963b; Floater 1997b] (essentially a discrete version of the Radó-Kneser-Choquet theorem). In its original form, it handles surfaces with disk topology and does not support cones. Some recent results [Gortler et al. 2006; Aigerman and Lipman 2015b,0; Bright et al. 2017] elegantly generalize the idea to other settings, but either not to arbitrary sets of cones, not to arbitrary topology, not using the piecewise linear Euclidean setting, or without similar guarantees on map existence.

Injectivity-preserving optimization. A variety of techniques have been presented for distortion optimization, e.g. [Schüller et al. 2013; Hormann and Greiner 2012; Rabinovich et al. 2017b;

Kovalsky et al. 2016b; Zhu et al. 2018b; Shtengel et al. 2017b]. Through line search techniques, barrier functions, and similar techniques they are able to guarantee preservation of injectivity – if initialized with an injective starting point. State-of-the-art techniques can handle large meshes efficiently and tolerate significant imperfections in the initial solution.

Seamless parametrization. A number of methods have been described for the construction of seamless parametrizations with prescribed cones [Kälberer et al. 2007; Bommes et al. 2009b,0; Myles and Zorin 2012,0; Ebke et al. 2016; Bright et al. 2017; Fu et al. 2015b; Chien et al. 2016; Hefetz et al. 2019]. Interestingly, but not surprisingly, they do not follow the above two step principle – as no general method for the first step (valid initialization) is known for the arbitrary-topology arbitrary-cones setting. Instead, they are typically based on optimization subject to non-convex constraints and, despite long development and practical importance, no concise sufficient conditions for success are known. The key issue is that there is no available way to construct an initial solution, and one cannot guarantee that the solver will itself find a way into the feasible region.

Only for certain special cases there are known solutions in this regard, e.g., for specific genus or specific cones [Aigerman and Lipman 2015b; Gu and Yau 2003], using more general non-piecewise-linear parametrization [Aigerman and Lipman 2016b], or requiring additional input [Tong et al. 2006]. Particular challenges are caused by the fact that the given surface discretization may not even admit a (elementwise linear) solution, i.e., systematic remeshing capabilities are needed in any reliable approach.

Cross-Field Guidance. Most often such parametrizations are generated and optimized guided by a cross-field or frame field on the surface [Vaxman et al. 2016]. Seminal works on cross-field guided parametrization are [Knupp 1995; Kälberer et al. 2007]. Important ideas for cross-field generation are presented by [Ray et al. 2008; Crane et al. 2010; Li et al. 2006; Bommes et al. 2009b; Ray et al. 2009]; many of these offer control over the fields' turning numbers.

Quadrangulation. The problem of surface quadrangulation with conforming elements and prescribed extraordinary vertices is closely related – state-of-the-art methods actually construct quadrangulations via seamless parametrization [Bommes et al. 2013c]. [Jucovič and Trenkler 1973] investigate the question of existence of such quadrangulations. The result is purely combinatorial and does not yield a surface parametrization. On an abstract level, we adapt some of the general ideas in this work as foundation of our approach to modify non-seamless into seamless parametrizations through map padding.

In the context of quadrangulation, our strategy of transitioning from an initial non-seamless parametrization to a seamless one is, in a sense, similar to modifying a non-conforming quadrangulation into a conforming one. This has been tackled by simple subdivision or more involved T-mesh simplification techniques [Myles et al. 2014] – however, at the expense of not always preserving the prescribed extraordinary vertices. Our modification technique, by contrast, always preserves exactly the prescribed cones.

Cone Choice. The choice of cones (and more generally guiding cross-fields, holonomy signatures) is an application dependent matter. Various approaches have been proposed for the selection of a cone configuration, for instance curvature-based (e.g. via cross-fields [Vaxman et al. 2016]), distortion-based [Kharevych et al. 2006b; Soliman et al. 2018b; Ben-Chen et al. 2008], or interactive [Ebke et al. 2016; Campen and Kobbelt 2014]. The problem of positioning cones such that conformal maps with these cones become seamless is addressed by [Chen et al. 2019,0].

General holonomy prescription. [Campen and Zorin 2017b] address a related problem, showing that for any admissible *holonomy signature* one can construct (also via conformal maps) a *seamless similarity* map adequate for constructing T-splines. A holonomy signature, in addition to prescribed cone angles, includes turning angles around homology loops. In contrast, we use a stronger notion of seamlessness, not allowing scale jumps across cuts, while not controlling global turning angles around homology loops (cf. Sec. 5.6) – however, they are of the form $k\pi/2$

(for *some* k) by our construction.

Guarantees. In special cases (restricted genus, restricted cone configurations) convex formulations can be used to reliably yield locally injective seamless parametrizations [Gu and Yau 2003; Gortler et al. 2006; Aigerman and Lipman 2015b]. Alternatively, additional user input like a surface partition may be exploited to ensure validity [Tong et al. 2006], or more general, non-piecewise-linear forms of parametrization may be employed [Aigerman and Lipman 2016b].

Recently, first methods have emerged that provide validity guarantees while supporting arbitrary genus and general cone configurations [Campen et al. 2019; Zhou et al. 2020]. In this sense, they offer control over *local* holonomy aspects. No *global* control over holonomy is provided, though. Therefore, when for instance aiming to generate a cross-field guided parametrization, while locally cones are reproduced, there may be global topological mismatches between the given cross-field and the constructed parametrization, for instance precluding proper alignment.

Some of the above methods for parametrization construction (such as [Bommes et al. 2009b; Bright et al. 2017]) offer full control over the resulting parametrizations' holonomy, but do not guarantee local injectivity. Those that guarantee local injectivity in a general setting (e.g. [Zhou et al. 2020; Campen et al. 2019; Myles et al. 2014]), in turn, do not offer full control over holonomy. The method of [Campen and Zorin 2017b] offers full holonomy control, albeit only for the broader class of seamless *similarity* parametrizations.

3 | Progressive Embedding

3.1 Introduction

Piecewise linear surface-to-plane maps, or parametrizations, are ubiquitous in computer graphics, geometry processing, mechanical engineering, and scientific visualization. Depending on the applications, the maps are required to exhibit different properties, most commonly, low distortion, local injectivity, and global bijectivity.

The last two properties are challenging to guarantee for discrete maps. Most algorithms with guarantees use Tutte embedding as a component. Tutte embedding is a construction that is guaranteed to create bijective mappings under minimal assumptions, if both domains are simply connected and the target planar domain is convex. However, the guarantee only holds if the computation is performed in arbitrary precision rather than floating point arithmetic, as it is commonly done. Failure due to floating point approximation is not as uncommon as one would assume, as the algorithm is likely to create an extreme variation of scale and aspect ratios in complex mapping cases. To quantitatively evaluate this issue, we computed Tutte embeddings on 2718 models (all the genus 0 models from Thingi10k [Zhou and Jacobson 2016]) using double precision, and observed 80 failures. To the best of our knowledge, this problem has not been addressed before in the literature.

This rate of failure is problematic for batch processing large geometrical collections (for example for processing geometric deep learning datasets) or when the embedding has to be computed

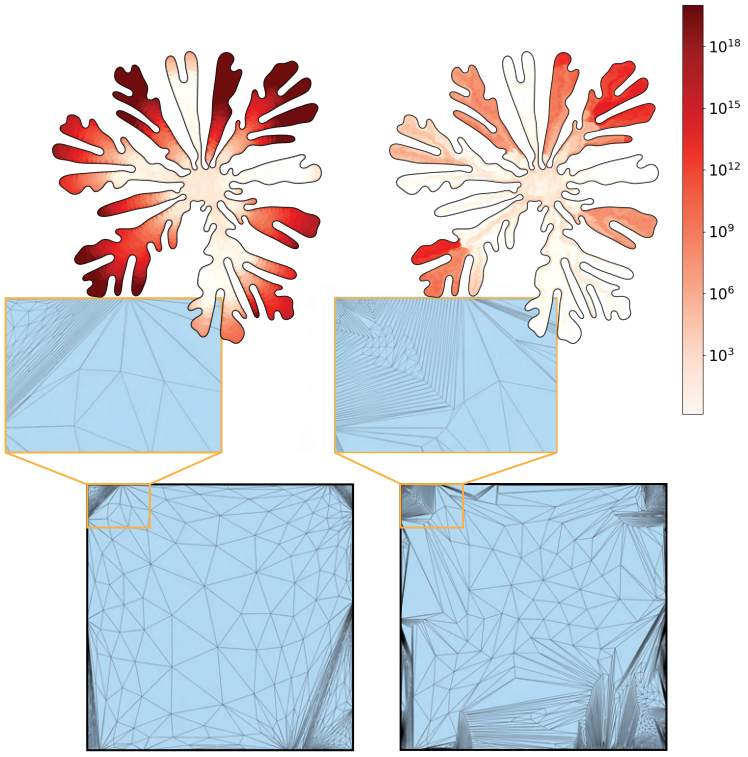


Figure 3.1: The Tutte embedding of this Hele-Shaw polygon (left) contains 46 flipped triangles, due to numerical rounding errors. Our progressive embedding (right) produces a valid embedding, without any inverted element and with lower distortion. The colors represent the distortion of the triangles, measured using the symmetric Dirichlet energy.

many times (for example in cross-parametrization [Kraevoy et al. 2003; Schreiner et al. 2004]). In these scenarios, a failure rate of 2.9% may not be tolerable, since it is not realistic to manually fix hundreds of problematic cases, and if failure happens on large meshes with millions of triangles it might not even be possible to fix them by hand. A simple solution to this problem is the use of multi-precision (or rational) arithmetic [Granlund 2018]: if enough bits are used to represent the mantissa and exponent of the floating point representation, Tutte embedding will succeed, since the solution of a linear system can be computed exactly. However, the result in high precision is not directly usable by downstream applications, and requires to be rounded (or "snapped" [Halperin and Packer 2002]) to floating point coordinates. This is a surprisingly challenging problem for which, to the best of our knowledge, no solution applicable to our setting

exists (Section 3.2.1).

Instead, we propose a *progressive* algorithm to directly generate an embedding using floating point coordinates. We start from an initial, possibly invalid, floating point planar parametrization, and we make it valid by collapsing all flipped and degenerate parts of the (possibly invalid) embedding produced, e.g., by a floating-point Tutte algorithm. We re-insert one vertex of the original mesh at a time, preserving the validity of the map at every step. This approach is inspired by [Schreiner et al. 2004], which proposes a progressive algorithm for computing cross-parametrizations based on progressive meshes [Hoppe 1996]. Our algorithm differs since we do not know a valid position for the inserted vertices, and we thus have to compute it as the vertices are added back. We provide a formal proof of correctness of our method in arbitrary precision (obtaining the same formal guarantees as Tutte embedding), and we practically demonstrate its superior robustness by parametrizing a large collection of 10k models.

Using our new embedding method and the matchmaker algorithm [Kraevoy and Sheffer 2004; Kraevoy et al. 2003] as a foundation, we develop an algorithm for mapping between multiply-connected domains with arbitrary constraints, supporting fully general self-overlapping domains as the target. We experimentally show that our algorithm is very robust, producing valid and distortion-optimized maps even for challenging cases where the original matchmaker algorithm fails due to numerical problems. We demonstrate the practical utility of our algorithm for UV mapping and quadrangulation applications.

To foster replicability of results and to maximize the practical impact of our algorithm, we also attach a reference implementation. `https://github.com/hankstag/progressive_embedding`

3.2 Progressive Embedding

3.2.1 Analysis of Tutte Embedding in Floating Points

We discuss in detail when Tutte embedding implemented in floating point may fail, and also show that straightforward solutions with off-the-shelves geometry processing tools do not solve these issues.

Tutte Embedding implemented in Floating Points We use the implementation of Tutte embedding in libigl [Jacobson et al. 2016], and apply it to all the 2718 genus 0 models of the Thingi10k dataset [Zhou and Jacobson 2016], after cleaning them up and improving their quality using TetWild [Hu et al. 2018], to ensure that no degenerate triangles are present. We also ensure that the meshes are 3-connected by refining them locally. For every model, we randomly pick and delete a triangle, and map the resulting boundary to an equilateral triangle. We compute the Tutte embedding, and check for flips using CGAL's exact floating point predicates [Brönnimann et al. 2018]. The check fails for 80 models, due to the numerical errors introduced in the mapping. In retrospect, this is not surprising since it is well-known that Tutte planar drawing may admit exponential area when drawing on integer grids. Two problematic cases are shown in Figure 3.2, where the embedding introduces a large variation of scale, and the flip occurs on triangles with small areas.

Multi-Precision Tutte Embedding with Snap Rounding. A straightforward way to address this problem is to increase the number of bits used in the floating point representation. We double the number of bits using the library MPFR [Fousse et al. 2007], which is directly integrated into Eigen, and can thus be used with the Tutte embedding in libigl with minimal code changes. With this setup, all the problematic cases are solved. However, the runtime is increased by around one order of magnitude, and, most importantly, the results generated cannot be rounded back to

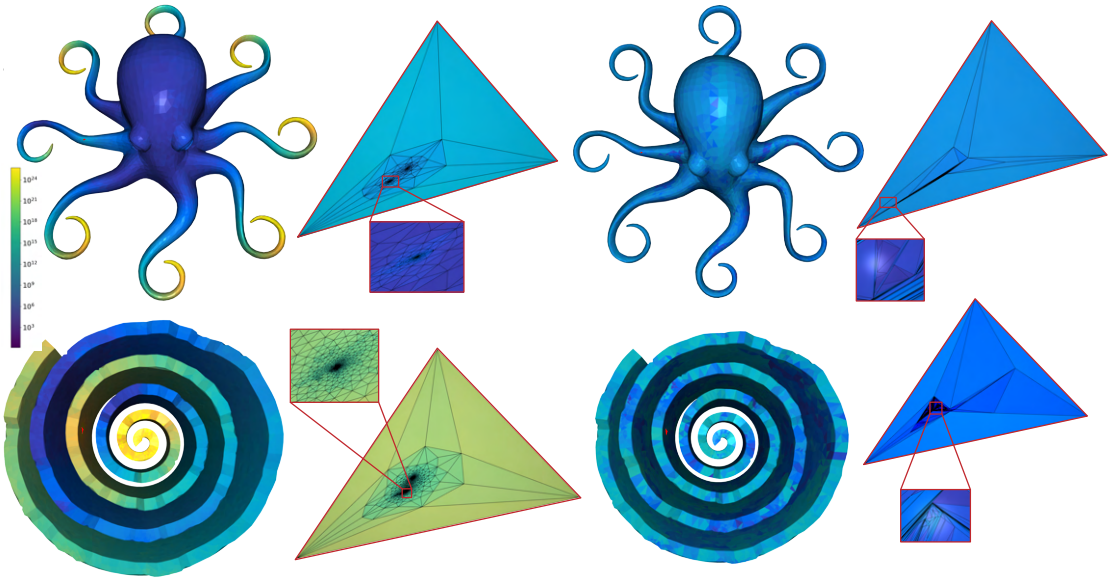


Figure 3.2: A selection of failed Tutte's embedding (left) and our bijective progressive embeddings (right). Note that the progressive embeddings have a much lower area distortion (colors).

floating point since trivial rounding introduces flips. Snap rounding [Packer 2018] could be used to avoid them, but it will collapse possibly large regions of the mesh: 6.3% of the vertices of the model shown in the bottom left of Figure 3.8 are collapsed when using a snap rounding resolution of 10^{-16} times the diagonal of the bounding box of the embedding.

Multi-Precision Tutte Embedding with Quality Optimization. The problem with rounding to floats is induced by the small triangles (and correspondingly small edges) which leads to flips after snapping. A possible way to address this issue is to use a mesh optimization algorithm, using multi-precision representation, before rounding to floats. We tested two approaches: (1) SLIM [Rabinovich et al. 2017a] adapted to run in multiprecision, and (2) minimizing the symmetric Dirichlet energy by moving one vertex at a time using coordinate descent [Hormann and Greiner 2012]. The first approach is prohibitively slow, due to the linear solve in high precision and the very small steps due to the elements with almost zero area. The second one succeeds on 57 models, but still fails on 23, even after 24 hours of running time.

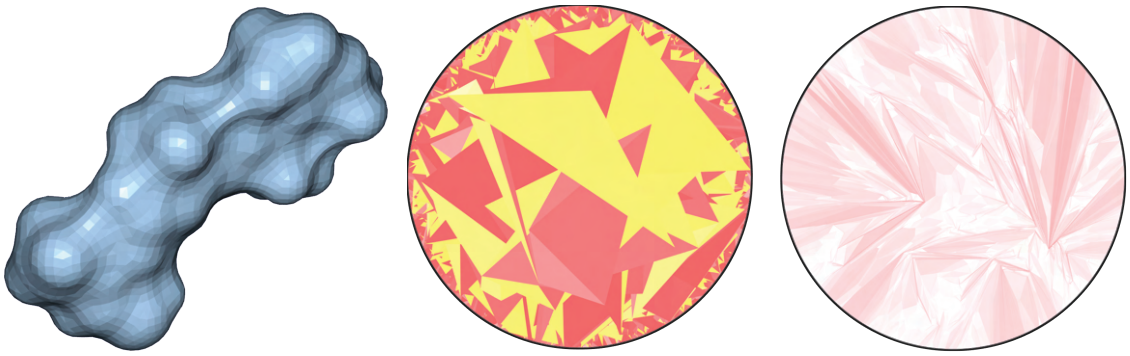


Figure 3.3: A progressive embedding (right) of the retinal model (left) is generated starting from a randomized initial parametrization (middle). The red color indicates the amount of isometric distortion, and yellow indicates inverted elements. Note that the model is cut open to have disk topology.

3.2.2 Progressive Embedding

Our approach draws on the ideas of progressive meshes [Hoppe 1996] and inter-surface mappings [Schreiner et al. 2004], which are, in turn, closely related to theoretical ideas from PL topology (e.g., [Hudson and Shaneson 1969]).

Our algorithm does not require Tutte embedding and can be used to construct an embedding from scratch or from a random initial mapping (Figure 3.3). It can be accelerated by using an existing, possibly *invalid*, embedding as a starting point. A triangle is invalid if its signed area is negative, or if its quality measure is below a threshold (we use the symmetric Dirichlet energy [Smith and Schaefer 2015] with respect to a canonical equilateral triangle whose area is the area of the target boundary polygon divided by the number of triangles and mark invalid if it is above $\tau = 1e20$). Using a quality measure in addition to signed area is important, since triangles with small, positive areas might cause numerical problems during the vertex insertion (Phase 2 below).

Starting from an invalid embedding, our algorithm (1) performs edge collapses until the simplified mesh has no invalid triangles (Algorithm 1), and (2) progressively inserts back each vertex in the same order (Algorithm 2), with feasibility of insertion ensured at each step.

Stage 1: Simplification. At this stage, we iteratively find an interior edge that can be collapsed until all invalid elements are removed from the initial embedding, or a single interior vertex is left

Algorithm 1: Collapse Invalid Triangles

Input : Planar mesh M
Output: Valid mesh M, and a recorded collapse sequence R

```
invalid_set = set of invalid triangles in M;
while invalid_set is not empty do
    if only one internal vertex left then
        Set to the barycenter of ∂ M and return
    for T ∈ invalid_set do
        for e ∈ T, internal(e) and link(e) do
            // Try only internal edges with link condition satisfied
            collapse(M, e) and record to R;
            Remove T from invalid_set;
            break ;                                   // Try next triangle
    if nothing got collapsed then
        // Expand the set with neighborhoods
        for T ∈ invalid_set do
            Add neighbors of T to invalid_set;
```

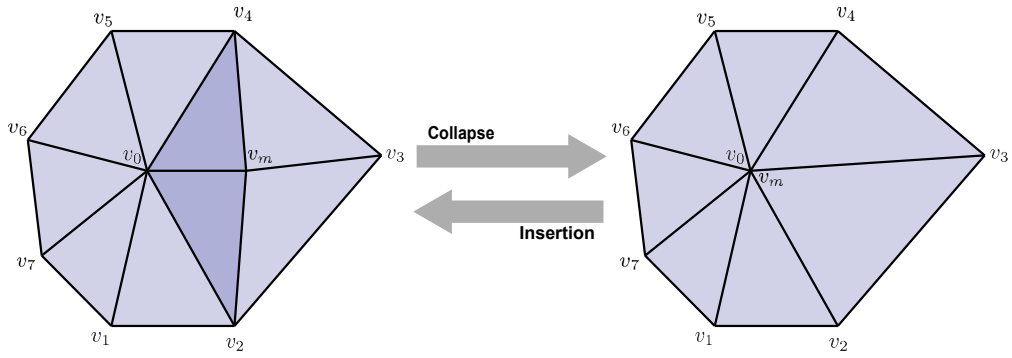


Figure 3.4: Collapsing v_m to v_0 and the corresponding fans of triangles.

(Algorithm 1). A theorem in [Mijatović 2003] and our Theorem 3.8 guarantees that a sequence of collapses reducing the mesh to a mesh with a single interior vertex can always be found; as the boundary embedding is convex, we can also always find a position for this vertex to create a valid embedding for the fully simplified mesh. The simplification algorithm starts by tagging all invalid triangles, and attempts to collapse all their edges. If this procedure is successful in eliminating all invalid triangles, the algorithm terminates, otherwise all the triangles adjacent to tagged triangles are tagged, and their edges collapsed. Note that we only allow edge collapses on internal edges to avoid changes to the boundary. This algorithm is guaranteed to terminate, since

Algorithm 2: Single Vertex Insertion

Input : Mesh M, v_m is to be split from v_0, with position p_0
$\mathcal{F}$ = neighboring faces of v_0 or v_m
V = adjacent vertices of v_0 in the valid sector
E = midpoints of edges in the link of v_0 in the valid sector
$\mathcal{P}$ = a map from the mesh vertices to 2D positions

Output: $\mathcal{P}$ with relaxed positions, along with newly assigned $\mathcal{P}(v_m)$

```
Loop
    foreach v ∈ V+E do
        // Backtracking line search, from p_0 towards P(v), until F all valid
        P(v_m) = linesearch (p_0, P(v));
        candidate_score = max_{f∈F} Energy (f);
        Record P(v_m) if candidate_score < ∞
    if Record is not empty then                          // Insertion succeeds
        Select P(v_m) with the minimum candidate_score.;
        Relax vertex positions with 10 iterations of local smoothing;
        break;
    else                              // Insertion fails, improve quality and try again
        Relax vertex positions with 50 iterations of local smoothing;
```

in the worst case it will tag the entire mesh and Theorem 3.8 ensures that at least one edge will be collapsible. If only one internal vertex is left, we move it to the barycenter of the boundary vertices (which is inside the convex boundary by construction, but might fail in degenerate cases, as discussed in Section 3.5). Once the algorithm terminates, the resulting simplified mesh has no inverted triangles and by Proposition 3.9, also has sums of triangle angles at each vertex equal to 2π.

Stage 2: Insertion. Starting from the valid embedding computed after Stage 1, we perform a sequence of splits reverting the collapses, while maintaining embedding validity at every step (Algorithm 2). Lemma 3.11 ensures that this is always possible (in infinite precision), as the area of each triangle after insertion is always positive, whenever the newly inserted points lie in the valid sector (Figure 3.5). The algorithm first computes candidate directions in the valid sector, then performs a flip-avoiding line search [Smith and Schaefer 2015] to find candidate positions $\mathcal{P}(v_m)$ for the newly inserted vertex v_m, such that the 1-ring neighborhoods are valid. In our

experiments, we use step length $\alpha = 0.8$, and cap the number of line search iterations to 75. If at least one candidate is found, the split is performed using the candidate resulting in a mesh minimizing the error measured as the maximum of the 1-ring energy. Since a candidate position always exist in infinite precision (Lemma 3.11), the only possible cause for not finding it is a lack of representation power in the floating point representation. We thus improve the quality of the mesh until a candidate is found.

This algorithm may still fail to find a candidate in degenerate configurations. However, we experimentally found that the constrained mesh smoothing is very effective at ameliorating this issue, keeping the mesh quality sufficiently high during the insertion to allow split operations to succeed (Figure 3.6). In all our experiments we only found one failure case, where the prescribed target boundary is a numerically degenerate triangle (Section 3.5). All our other experiments, even on a large data set and with complex boundary conditions (Section 3.4) were successful.

Local Smoothing. To improve the quality of the map in the insertion step (Algorithm 2, line 8 and 11), we minimize the symmetric Dirichlet energy [Smith and Schaefer 2015], optimizing one vertex position at a time using Newton iterations, similarly to [Hormann and Greiner 2012; Labsik et al. 2000; Fu et al. 2015a]. We favor this local approach since it is more robust to low quality elements, which would otherwise badly affect both the numerical stability and the step size of global optimization methods. Since our goal is to improve the minimal quality of the mesh, we minimize the symmetric Dirichlet energy only for the invalid triangles (using reference shape as an equilateral triangle with area equal to the average of triangles in the 2D domain), and use the scaffold energy [Jiang et al. 2017] (i.e. we use the element itself as the reference triangle for the symmetric Dirichlet energy) for the valid ones, which allows them to move more freely. In our experiments, the local smoothing is performed for 10 iterations after every insertion step. If a valid insertion candidate cannot be found, we keep improving the quality with batches of 50 smoothing iterations until a candidate is found (Algorithm 2 Furthermore, at each smoothing

phase, we perform a greedy coloring of the edge graph [Kucera 1991], and the vertices inside each color are optimized in parallel.

3.3 MATCHMAKER++

The computation of locally injective maps is important in geometry processing (Section 2.1), with the majority of the methods focusing on efficient and scalable quality optimization. However, few method guarantees positional constraints: the notable MatchMaker algorithm [Kraevoy et al. 2003] reduces the problem to a number of convex planar embeddings, which are computed with Tutte's algorithm. However, as we observe in some cases (Section 3.4.2), such embeddings can be numerically challenging. Replacing Tutte embedding with progressive embedding enables matchmaker to robustly compute maps with very challenging configurations of constraints. In this Section, we describe an extension of [Kraevoy et al. 2003] that (1) makes use of progressive embedding to increase robustness, and (2) supports weakly self-overlapping polygons as co-domains [Weber and Zorin 2014].

OVERVIEW. Combining our progressive embedding algorithm and the matchmaker algorithm, we describe an algorithm for solving the following problem: *Given a simply-connected 3d mesh, equipped with with a set of user-defined hard positional constraints at vertices, compute a valid piecewise-linear parametrization, such that (1) the map is valid in the following sense: there are no flipped triangles, and for each vertex, the map restricted to the one ring of triangle of that vertex is bijective, unless it is a singular boundary vertex, as defined below and (2) the parametrization bitwise exactly satisfies the user-defined positional constraints.* We tackle this in three steps: we decompose the target domain into convex polygonal subdomains, *match* these domains to the subdomains of the source domain, compute an initial bijective map by stitching progressive embeddings for each subdomain, and final globally optimize the mapping distortion.

User input. We distinguish between two cases, chosen by the user (1) the required map is a global embedding, (2) the map is an immersion. For the first case, the constraint specification is more flexible: the user only has to provide a set of point or line constraints. For the second case, the target domain is not a subset of the plane, but rather, an everywhere flat surface with overlaps. We require the user to prescribe constraints for the whole boundary of the polygon to define the target domain unambiguously (some parts may be marked as movable, but an initial position is needed) and to provide a path connecting each point or line constraint to the boundary, which allows to define its location on the target surface implied by the boundary specification. In this case, target boundary polygon has to be *weakly self-overlapping* [Weber and Zorin 2014], otherwise, the map does not exist.

Phase 1: Subdivision of the Target Domain. In the global embedding case, the target domain is generated by triangulating the bounding box of the input with Triangle [Shewchuk 1996]. In the second case, the self-overlapping domain is triangulated using a modification of the Shor-Van Wyck algorithm [Shor and Van Wyk 1992], described in [Weber and Zorin 2014]. In both cases we ensure that the hard positional constraints are vertices of the triangulation. We then merge triangles into convex polygons in a greedy manner, by dropping edges if the resulting subdomains are convex. While this merging step is optional (the algorithm works also without it) it reduces the number of subdomains and vertices, making the next steps more efficient. In the case when an immersion is computed, by construction, it will be an embedding on subdomains computed starting from the Shor-Van Wyck triangulation.

Phase 2: Path Tracing on the Original Mesh. After the target domain is subdivided into convex polygons, we *match* this decomposition to the input mesh. The goal is to find non-intersecting paths connecting each of these pairs in 3D mesh, subdividing the whole mesh into same number of patches.

At the tracing stage, we perform a reordering of the paths to make sure that no previously

traced path will block the future ones (Figure 3.7). In the case of embedding, the algorithm finds paths that connect all positional constraints to the boundary without creating additional loops (than the existing boundary). In practice, these paths are found by dropping one segment on the boundary, and then grow the minimum spanning tree (over the edges of the polygonal mesh) from the incomplete boundary loop. We first trace the paths on the minimum spanning tree and then the remaining ones, connecting the boundary to the constraints. The correctness of this procedure can be found in [Praun et al. 2001].

For the tracing of each path on the surface, we follow [Kraevoy et al. 2003] to find the shortest path connecting two endpoints, and add Steiner points on the edges if no path, not intersecting other paths, can be found.

Phase 3: Bijective Mapping. After establishing a correspondence between each patch of the 3D mesh and a convex polygon in the target domain, we can first subdivide the 2D paths in the target domain to match the number of vertices on the corresponding 3D path to obtain the one-to-one correspondence between them. We observe that up to this point, the algorithm is largely combinatorial (while some vertices are inserted on edges, their geometric position is trivially determined and is very unlikely to result in numerical problems; none were observed in our experiments). At this point, the map is defined for boundaries of the subdomains corresponding to the convex subdomains in the target. Next, we extend the map to the interior of each region using our progressive embedding algorithm (Section 3.2.2). Notice that when the mesh patch is not 3-connected, we need to split the edges with two endpoints on the boundary.

Phase 4: Quality Optimization. The map obtained in the previous steps is valid, according to the definition of the weakly self-overlapping map [Weber and Zorin 2014]. Therefore, its quality can be optimized using any locally injective map improvement algorithm (Section 2.1). We opt for [Rabinovich et al. 2017a], since it is efficient for large models and the implementation is readily available [Jacobson et al. 2016]. The implementation is modified to support hard positional

Table 3.1: Statistics of the input and output meshes in the planar embedding test (Section 3.4.1). From left to right: Name of the dataset, number of vertices, number of faces, number of invalid elements (positive area, but with energy above 1e20) after Tutte embedding, number of flipped elements after Tutte embedding, progressive embedding (Section 3.2) running time in seconds.

Name	#V	#F	#invalid	#flipped	PE(s)
Octopus	5034	10063	2351	524	245.1
Swirl	11754	23503	9317	638	2273.1
Deer	8720	17434	15728	7831	3916.8
Rabbit	7253	14500	8743	4233	1198.6
HeleShaw	3505	5355	437	46	62.1
Retinal	3791	7282	3533	3533	95.0
Arch	973	1941	790	270	21.9
Propeller	787	1569	484	70	11.6

constraints, by eliminating the corresponding variables.

3.4 Results and Discussion

We implemented our algorithm in C++, using Eigen [Guennebaud et al. 2010] for linear algebra, and libigl [Jacobson et al. 2016] for geometry processing and visualization. The reference source code, the data used, and the scripts to reproduce the results are attached in the additional material. The timings and statistics for the datasets shown in the paper are summarized in Table 3.1 and Table 3.2.

We first present results computed using only our progressive embedding (Section 3.4.1), and then demonstrate the generation of low distortion, locally bijective maps created with our extension of MatchMaker (Section 3.4.2).

3.4.1 Progressive Embedding

Planar Embedding for the Thingi10k Dataset [Zhou and Jacobson 2016]. By computing Tutte's embedding for the genus-zero models in 2718 surface mesh models on a triangle boundary, we observed there are 80 cases where the generated parametrization has flipped elements due to floating point rounding errors. Using our progressive strategy, we are able to fix all failed cases.

Table 3.2: Statistics of the input and output meshes of the MatchMaker++ test (Section 3.4.2). From left to right: Name of the dataset, number of vertices, number of faces, number of invalid elements (positive area, but with energy above 1e20) after Tutte embedding, number of flipped elements after Tutte embedding, progressive embedding (Section 3.2) running time in seconds, and MatchMaker++ (Section 3.3) running time in seconds.

Name	#V	#F	#invalid	#flipped	PE(s)	MM++(s)
Fertility	16508	33028	0	0	NA	582.4
3 holes	7440	14886	0	0	NA	107.4
Robot Cat	4117	7512	0	0	NA	0.8
Aircraft	2523	4656	0	0	NA	0.5
Twirl	5562	10402	0	0	NA	1.1
Filigree	49872	100000	32	0	72.4	30.8
Botijo	43786	83788	0	0	NA	3.9
Beetle	20619	39276	0	0	NA	1.1
Casting	21236	39438	67	40	27.4	1.3
Oil pump	54135	103778	5	0	2.3	4.8

A selection of the parametrization results are shown in Figure 3.2.

Integration with OptCuts [Li et al. 2018]. OptCuts is a joint optimization method to create UV seam from a 3D model, balancing seam length and parameterization quality. It relies on a valid initialization, which for genus 0 model, is compute through randomly cutting two adjacent edges as seams, then flatten it on the plane with Tutte embedding. In Figure 3.8, we show two examples where this initialization fail. Both models can be processed if progressive embedding is used instead of Tutte embedding, allowing OptCuts to proceed and optimize the UV map.

Mapping an Hele-Shaw Polygon to a Square. Hele-Shaw flow is a two-dimensional Stokes flow of mixing liquids between two parallel flat surfaces separated by a small gap. In Figure 3.1, we show an example mesh generated using the Hele-Shaw simulation proposed in [Segall et al. 2016]. One way to compute a bijective map of the interior of the polygon between different frames is a cross-parameterization using a square as the common domain, with no internal constraints. Tutte embedding fails in this case, introducing 46 flipped faces (Figure 3.1, left), while progressive embedding produces a valid map with lower distortion (Figure 3.1, right).

3.4.2 MATCHMAKER++

SELF-OVERLAPPING LOCALLY-INJECTIVE MAPS. By introducing Shor Van Wyck algorithm into the matchmaker pipeline, we are able to mapping a surface mesh with disk topology to self-overlaping boundaries as in [Weber and Zorin 2014]. Similarly to [Weber and Zorin 2014] our algorithm can generate locally-injective, self-overlapping parametrizations (Figure 3.9), which are commonly used by quadrangulation algorithms [Bommes et al. 2012].

COMPARISON WITH [KOVALSKY ET AL. 2015]. We parametrized the global parametrization benchmark introduced in [Myles et al. 2014], using the seams in the obj files, and fixing in random positions 3 random points of each mesh. This is a challenging task, since the random constraints introduce a large distortion. Our method succeeded on all 102 models: a selection of the most challenging ones is shown in Figure 3.10. We also run the same experiment using the most recent projection method [Kovalsky et al. 2015] (which is one of the few methods that supports similar constraints without requiring a fully specified target domain), using LSCM [Lévy et al. 2002b] as an initial guess. The method failed on 28 models over 102 (27%). We show three failed cases using their method with flipped elements in the output, and the quality is considerably lower than our approach, as shown in Figure 3.11. Note that this is a comparison that favours our method, since we are allowed to remesh the map, while [Kovalsky et al. 2015] preserves the original connectivity.

STRESS TEST. To further evaluate the robustness and applicability of our algorithm, we performed an additional stress test, by parametrizing the 102 models of [Myles et al. 2014] into a planar space filling curve, and adding 3 random positional constraints. These experiments push the algorithm to the limit: MatchMaker fails on 5 if Tutte embedding is used, while it succeeds in all cases, producing bijective maps exactly satisfying the hard positional constraints, with progressive embedding (Figure 3.12).

3.5 Limitations

We introduced a robust algorithm to compute planar embeddings, and demonstrated its practical utility in common geometry processing tasks. Our algorithm is provably correct in infinite precision and is designed to work robustly with floating point coordinates: unfortunately we cannot guarantee that an output is produced in the latter case since a solution of the local point placement problem might not exist. Consider the example in Figure 3.13: the bounding box of the triangle has short sides (the difference between the floating point coordinate representation is only in the least significant bit of the mantissa). Assume that our algorithm needs to split off a vertex from the vertex with numerically flat angle A, placing the resulting point in the interior. In this situation, our algorithm fails, since the average of the coordinates (in floating point) of the boundary triangle does not lie inside the triangle due to numerical rounding.

Except for this extreme case, we have not observed any other failure cases for our algorithm, which produced robustly thousands of embeddings, and, when paired with matchmaker, enables the robust generation of constrained locally injective maps.

3.6 Proofs

The proof of the existence of the collapse sequence for two-dimensional manifold meshes can be found, e.g., in [Mijatović 2003], where it is derived from the *shellability* of two-dimensional manifold meshes homeomorphic to a disk (i.e., the possibility of removing triangles one-by-one, keeping the topology of the remaining part of the mesh unchanged), and make use of a specific composition of Pachner moves equivalent to edge collapse. We present a different proof, based on proving the existence of a collapsible edge, which is aligned with the structure of our algorithm and helps us to show the existence of the inverse vertex split sequence.

3.6.1 Existence of the Collapse Sequence

We assume that the input mesh connectivity (V_0, F_0) is manifold, i.e., each edge is shared by no more than two triangles, and the triangles incident at a vertex can be arranged in a sequence so that two sequential triangles share an edge. For interior vertices, the sequence is circular, i.e. the first and last triangles also share an edge. With the topology of a 2D disc, the graphs of edges of such meshes are *planar* i.e., can be embedded in the plane, with positions $P_0 = \{p_i \in \mathbb{R}^2\}$ assigned to vertices v_i. By the Fáry's theorem, [Fáry 1948] there is a straight-edge embedding of this graph in the plane with non-intersecting edges (*Fáry embedding*). In subsequent lemmas, we use geometric images of vertices and edges under this embedding. Only the existence of this embedding, but not the specific construction, is used to prove the existence of the collapse sequence.

The following sequence of lemmas focuses on the one-ring neighborhood of an interior vertex, and shows that at least one of the adjacent edges satisfies the link condition. This observation further leads to Theorem 3.8: a valid sequence of edge collapses can be used to reduce the mesh to a mesh with a single interior vertex. A vertex of the mesh is *interior* if it does not lie on the boundary, and an edge is *interior* if its two endpoints are interior vertices.

Definition 3.1. An interior edge $\overline{v_i v_j}$ satisfies the *link condition* if $|N_i \cap N_j| = 2$, where N_i is the set of the adjacent vertices of v_i.

Lemma 3.2. *Let v_0 be an interior vertex of degree d (Figure 3.14). We enumerate its neighbors counterclockwise around the vertex (using Fáry embedding), denoting them $v_1, v_2, \ldots, v_d$. Assume $\overline{v_0 v_1}$ violates the link condition, i.e., $N_0 \cap N_1$ contains a vertex v_k, $k = \min(N_0 \cap N_1 \setminus \{2, d\})$. (1) If the triangle $\Delta v_0 v_1 v_k$ is oriented counterclockwise, then the set N_i, consisting of adjacent vertices of v_i,lies within $\Delta v_0 v_1 v_k$, for any $1 < i < k$. (2) If $\Delta v_0 v_1 v_k$ is oriented clockwise, then N_i is within $\Delta v_0 v_1 v_k$ for $k < i < d$.*

Proof. Without the loss of generality, consider $\Delta v_0 v_1 v_k$ orients counterclockwise. Consider the segment $\overline{v_0 v_i}$, for $1 < i < k$. The half-line starting at v_0 and containing this segment is between half-lines containing v_1 and v_k, because the vertices were numbered counterclockwise. Therefore the half-line contains a point in the interior of $\Delta v_0 v_1 v_k$, by continuity. If v_i is outside or on the boundary of $\Delta v_0 v_1 v_k$ then the half-line connects an interior and non-interior point different from v_0, and intersects $\overline{v_1 v_k}$. which contradicts the assumption on the embedding. Thus, v_i is in the interior of $\Delta v_0 v_1 v_k$. Similarly, all points in N_i are either in the interior of $\Delta v_0 v_1 v_k$, or on its vertices, as the edges of the embedding do not intersect except at vertices. □

Without loss of generality, we assume that the first case of the lemma and take a closer look at $\Delta v_0 v_1 v_k$. Intuitively, one can think of an edge that violates the link condition as having two endpoints which are connected to (at least) three common vertices. Therefore, on one of the sides of the edge, there would be at least two vertices connected to it. The next lemma establishes the fact that if a sequence of edges violates the link condition, then the "lower"(smaller indices) side of the edge always has only one vertex connected to its endpoints.

Lemma 3.3. *Suppose an edge $\overline{v_0 v_1}$ violates the link condition, and k is defined as in Lemma 3.2. Suppose, W.L.O.G., $\Delta v_0 v_1 v_k$ is oriented counterclockwise, and let $1 < i < k$. If additionally for all $1 < t < i$, $\overline{v_0 v_t}$ violates the link condition, then v_{i-1} is the only vertex with index in the range $1 \leq t \leq i-1$ connected to v_i.*

Proof. We prove the Lemma by induction. The base case, $i = 2$ the proposition clearly holds. Suppose for all $i \leq j$ holds. Since $\overline{v_0 v_j}$ violates the link condition, $N_0 \cap N_j$ contains v_{j+1} and v_n for some $j+1 < n \leq k$ (see Figure 3.15), by the inductive assumption. v_{j+1} is in the interior of $\Delta v_0 v_j v_n$ by Lemma 3.2. For $m \leq j-1$, v_m is outside $\Delta v_0 v_j v_n$, because the half-line $\overline{v_0 v_j}$ is between $\overline{v_0 v_m}$ and $\overline{v_0 v_j}$ by the choice of numbering. It follows that in order to connect v_{j+1} to a previous vertex v_m $\overline{v_{j+1} v_m}$ would have to intersect the boundary of $\Delta v_0 v_j v_n$, which contradicts that fact that we are using an intersection-free Fáry embedding. This proves the induction step. □

We conclude that under the assumptions of Lemma 3.2, first case, $v_i, 1 < i < k$ are interior vertices in the triangle $\Delta v_0 v_1 v_k$, thus interior vertices of the mesh. Then $\overline{v_0 v_2} \dots \overline{v_0 v_{k-1}}$ are interior edges. The next lemma shows that at least one of them satisfies the link condition

Lemma 3.4. *Following the first case in Lemma 3.2. If for all n < k-1, $\overline{v_0 v_n}$ violates the link condition, the interior edge $\overline{v_0 v_{k-1}}$, satisfies the link condition.*

Proof. By definition, $N_0 \cap N_{k-1}$ is not empty. By Lemma 3.3, the only vertex with index less than $k-1$ contained in N_{k-1} is v_{k-2}. On the other hand, the only remaining vertex of N_0 with index greater than $k-1$ inside $\Delta v_0 v_1 v_k$ is v_k. So we have exactly two vertices in $N_0 \cap N_{k-1}$, i.e., $\overline{v_0 v_{k-1}}$ satisfies the link condition. □

Definition 3.5. A *fan of triangles* $\mathcal{F}(v_0; v_1 \dots v_{d+1})$, centered at v_0, with v_i enumerated counter-clockwise around v_0, is a sequence of non-repeating triangles $\{\Delta v_0 v_i v_{i+1}) | i = 1 \dots d\}$. A fan is *closed*, if $v_{d+1} = v_1$, otherwise it is *open*.

Definition 3.6. Given a triangulation of a polygonal planar domain, with two interior vertices v_0, v_m, whose neighbors are N_0 and N_m, a *collapse operation* from v_m to v_0 connects all vertices $v \in N_m \setminus N_0$ to v_0, and removes v_m with incident edges. A collapse operation is *valid* if $\overline{v_0 v_m}$ satisfies the link condition, and neither of the end points is a boundary vertex.

To define a collapse operation in a reversible way, in addition to specifying the pair of vertices, we define a fan $\mathcal{F}(v_0; v_1 \dots v_k)$ in the mesh obtained after the collapse. The vertices $v_1 \dots v_k$ are the vertices that were connected to v_m before the collapse. In other words, we record a collapse operation, transforming the mesh (V_i, F_i) to (V_{i+1}, F_{i+1}), as the pair $C_i = (v_m, \mathcal{F}(v_0; v_1 \dots v_k))$, where v_m is the removed vertex in V_i, and $\mathcal{F}$ is a fan of triangles in F_{i+1}.

Lemma 3.7. *A 3-connected and planar mesh, is still 3-connected and planar after any interior edge collapse $C = (v_m, \mathcal{F}(v_0; v_1 \dots v_k))$.*

Proof. The link condition ensures that the mesh remains manifold after an edge collapse [Dey et al. 1999]. 3-connectedness of a triangle mesh is equivalent to the requirement that no two boundary vertices are connected by an interior edge. As no collapses involving boundary vertices are allowed, if there are no such edges before the collapse, no such edge may appear after the collapse: the only new edges connect vertices of the fan of v_m to v_0, which is interior. □

Theorem 3.8. *If the edge graph of a mesh (V, F) is planar and 3-connected, there is always an edge that can be collapsed to obtain a planar and 3-connected mesh with one less vertex, unless there is only one interior vertex left.*

Proof. Suppose no edge in (V, F) can be collapsed. This means that either there are no edges with two interior endpoints, or all such edges violate the link condition. But by Lemma 3.4, the second option is not possible. If there are no edges connecting two interior vertices, then all edges incident at interior vertices have the other endpoint on the boundary. Then all edges in the link of an interior vertex have two endpoints on the boundary. By 3-connectedness, these edges should be boundary edges. Therefore, the link of each interior vertex forms a complete boundary loop. As we assume the mesh to be simply connected, then there is only one boundary loop. So the whole boundary has to coincide with the link of any interior vertex, from which it follows that there is only one. □

We remark that when there is only one interior vertex left, if the boundary vertices v_i, $i \geq 1$, are assigned positions p_i, so that they form a star-shaped simple polygon, there is a position (within the interior of the kernel of the boundary) p_0 for the remaining interior vertex v_0 that results in a valid straight-edge embedding.

As a result of sequentially collapsing edges, we obtain a sequence $(V_i, F_i, C_i), i = 1, 2, \ldots, k$ with the following properties: $|V_i| = |V_{i-1}| - 1$, (V_k, F_k) is a valid triangulation, boundary vertices are the same for all V_i, and C_i is a valid collapse.

Proposition 3.9. *Suppose the vertices v_i of the disk-topology manifold mesh (V, F) are assigned*

parametric positions p_i in the plane, and the map is bijective on the boundary so that the triangles all have positive orientation. Then the sum of the angles of triangles incident at an interior vertex is 2π.

Proof. Assign, e.g., unit length to all edges of the mesh; this associates a a surface M with the mesh, with each combinatorial triangle corresponding to an equilateral triangle. Then the positions p_i define a PL map from M to the plane. By Theorem 1 from [Lipman 2014], this map is globally bijective; the statement of the proposition immediately follows. □

3.6.2 Vertex split

Definition 3.10. Let (V, F) be a mesh with a valid straight-edge embedding in the plane given by vertex positions P. Consider a closed fan of triangles $\mathcal{F}(v_0; v_1 \dots v_{d+1})$ centered at an interior vertex v_0, with $v_{d+1} = v_1$, and an open sub-fan $\mathcal{F}(v_0; v_1 \dots v_k)$. *Vertex split* introduces a new vertex v_m with a position p_m, $\mathcal{F}(v_0; v_1 \dots v_k)$, replaces it with a fan $\mathcal{F}(v_m; v_1 \dots v_k)$, and adds triangles $\Delta v_0 v_1 v_m$ and $\Delta v_0 v_m v_k$. We denote such a split S by $(v_m, p_m, \mathcal{F}(v_0; v_1 \dots v_k))$.

A split $S = (v_m, p_m, \mathcal{F}(v_0; v_1 \dots v_k))$, in terms of connectivity modification, is the inverse of a collapse $C = (v_m, \mathcal{F}(v_0; v_1 \dots v_k))$: the connectivity of the mesh obtained by applying the split is identical to the mesh that the collapse was applied to.

The following lemma establishes that we can perform a split reversing any collapse while maintaining the validity of the embedding, if the initial embedding is valid.

Lemma 3.11. *Consider a fan of triangles $\mathcal{F}(v_0; v_1 \dots v_k)$ (Figure 3.5), with positive signed areas $\{A(\Delta v_0 v_{i-1} v_i)\}_{1<i\le k}$ and with angles of triangles incident at v_0 summing up to 2π. The kernel of the fan has a non-empty interior. Then a new vertex position p_m corresponding to a new vertex v_m located in the interior of the fan, can be split off p_0, so that $\min_{1<i\le k} A(\Delta v_m v_{i-1} v_i) > 0$, $A(\Delta v_0 v_1 v_m) > 0$, $A(\Delta v_0 v_m v_k) > 0$, and angles of triangles in both resulting fans at v_0 and v_1 sum up to 2π.*

Proof. Define a function $f(p_x) = \min_i A(\Delta v_x v_{i-1} v_i)$. This is a continuous function of the coordinates p_x of the point v_x. Because $f(p_0) > 0$, there is a disk $B(p_0, \varepsilon)$, of radius $\varepsilon > 0$, such that $f(p_x) > 0$ for any $p_x \in B(p_0, \varepsilon)$, i.e. for all i, $A(\Delta v_m v_{i-1} v_i) > 0$, if we pick p_m inside $B(v_0, \varepsilon)$. Suppose we initially place p_m at p_0, with new triangles added as a result of the split having zero angles at v_1 and v_k. We note that the for each of v_0 and v_1 in this degenerate configuration the angles of incident triangles sum up to 2π. The angles of triangles also change continuously as functions of vertex position p_x, so does their sum. On the other hand, if each triangle remains positively oriented ($A(\Delta v_m v_{i-1} v_i) > 0$), then the sum of the angles can only change discretely, and has to be of the form $2\pi n$, $n \in \mathbb{Z}$ (n-fold cover). We conclude that n has to remain one, as it is one for the initial position.

Consider the intersection C of the half-planes bounded by lines containing $\overline{p_0 p_1}$ and $\overline{p_0 p_k}$ (for each segment, we choose the half-line on the side of the interior of the fan). If $p_m \in C \cap B(p_0, \varepsilon)$, then $A(\Delta v_0 v_1 v_m) > 0$, $A(\Delta v_0 v_m v_k) > 0$ also hold. □

The following theorem is a straightforward application of Lemma 3.11.

Theorem 3.12. *Suppose we have a sequence of valid collapses (V_i, F_i, C_i), $i = 0 \dots N-2$, where $N = |V_0|$, the number of interior vertices in the initial mesh, all (V_i, F_i) $i = 0 \dots N-1$ are 3-connected planar, and the last mesh (V_{N-1}, F_{N-1}) with a single interior vertex has a valid straight-edge embedding in the plane with vertex positions P_{N-1}. Suppose $C_i = (v_m^i, \mathcal{F}(v_0^{i+1}, v_1^{i+1} \dots v_k^{i+1}))$.*

Then the sequence of vertex splits S_i that are inverses of C_i, results in a valid straight-edge embedding of (V_0, F_0), given by vertex positions P_0.

Proof. V_{N-1}, F_{N-1} with positions P_{N-1} is valid by assumption. Each step of vertex split with S_i results in a straight-edge embedding of (V_i, F_i) by Lemma 3.11. By induction, the embedding of (V_0, F_0) obtained by the sequence of splits reverting the sequence of collapses is a straight-edge embedding. □

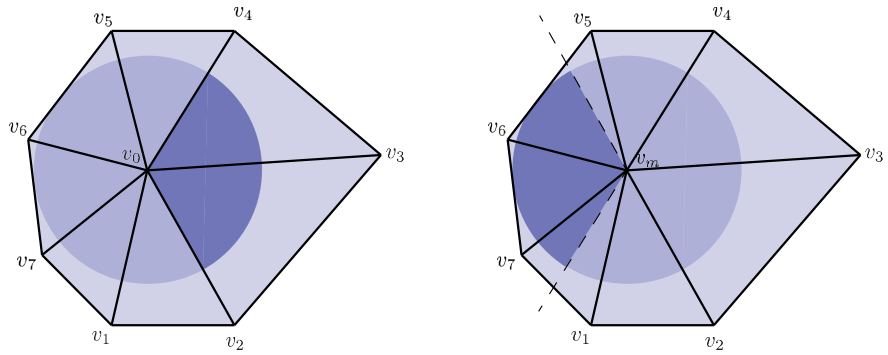


Figure 3.5: The two admissible insertion positions from Lemma 3.11. The dark region on the left shows the valid positions for v_m while fixing v_0. The right case is the opposite. Our algorithm opts for the left case for stability, since the calculation of the valid sector in the right case involves intersection of the prolonged edge (dashed lines) and the 1-ring neighbors. We pick the valid sector as the one that has an inner angle sum smaller than π.

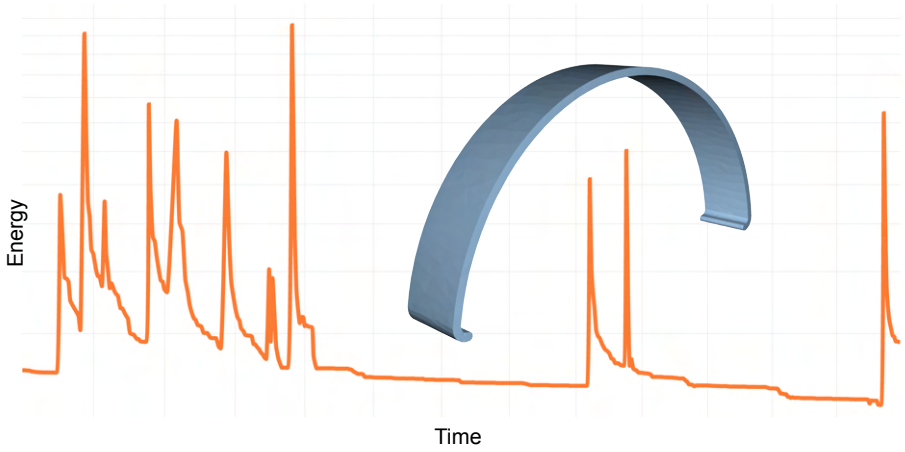


Figure 3.6: Max of Symmetric Dirichlet energy per triangle at the insertion stage of the arch model. Every vertex insertion can decrease the local quality of the mesh, which is then restored using smoothing. Every peak in the energy graph corresponds to a vertex insertion.

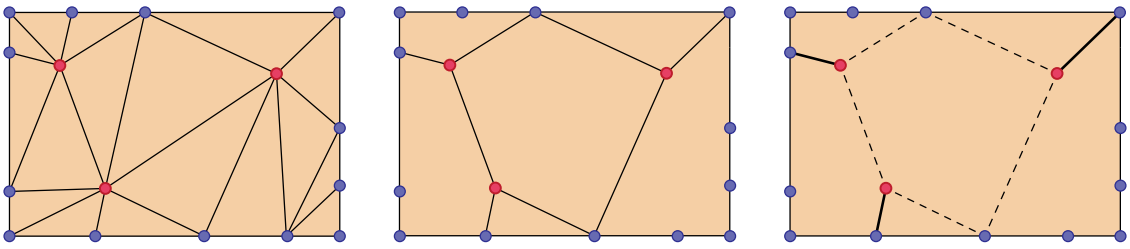


Figure 3.7: Starting from a triangulation generated from only boundary segments and internal constraint points (left). Instead of treating triangles as sub-domains as in [Kraevoy et al. 2003], we merge triangles to convex polygons (middle). Then we find paths (bold, right) connecting constraint points to the boundary without new cycles, and prioritize their tracing.

Figure 3.8: Three UV maps generated by OptCuts [Li et al. 2018] using an initial embedding created by our algorithm. OptCuts fails to process both models if Tutte embedding is used instead.

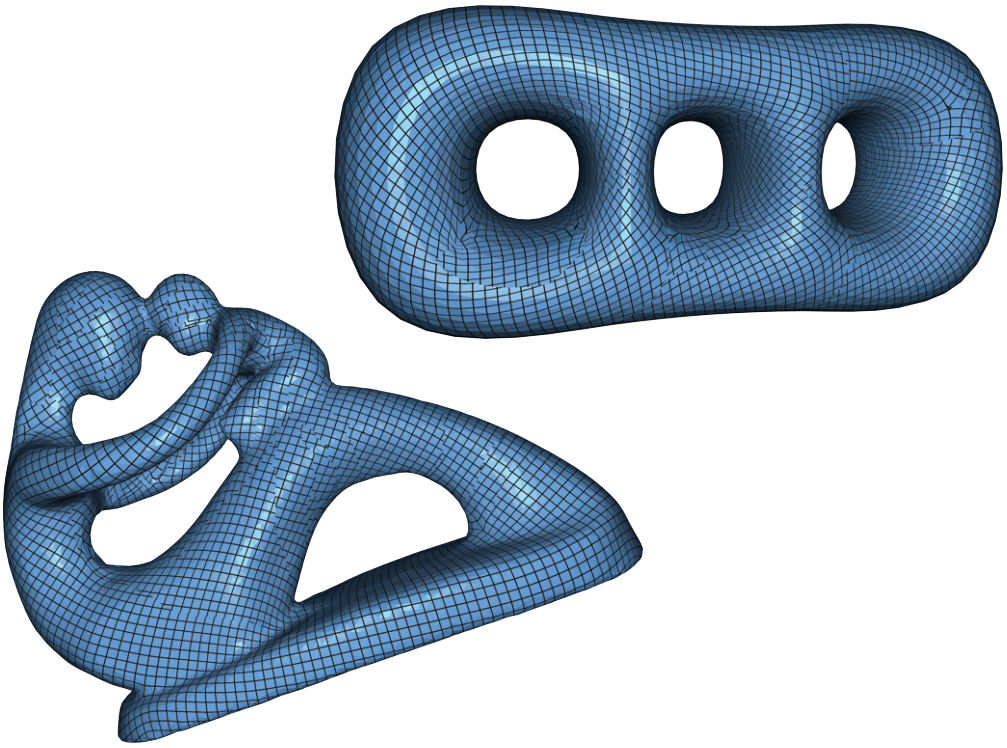


Figure 3.9: Two seamless maps with hard positional constraints and fixed boundaries are generated by our algorithm.

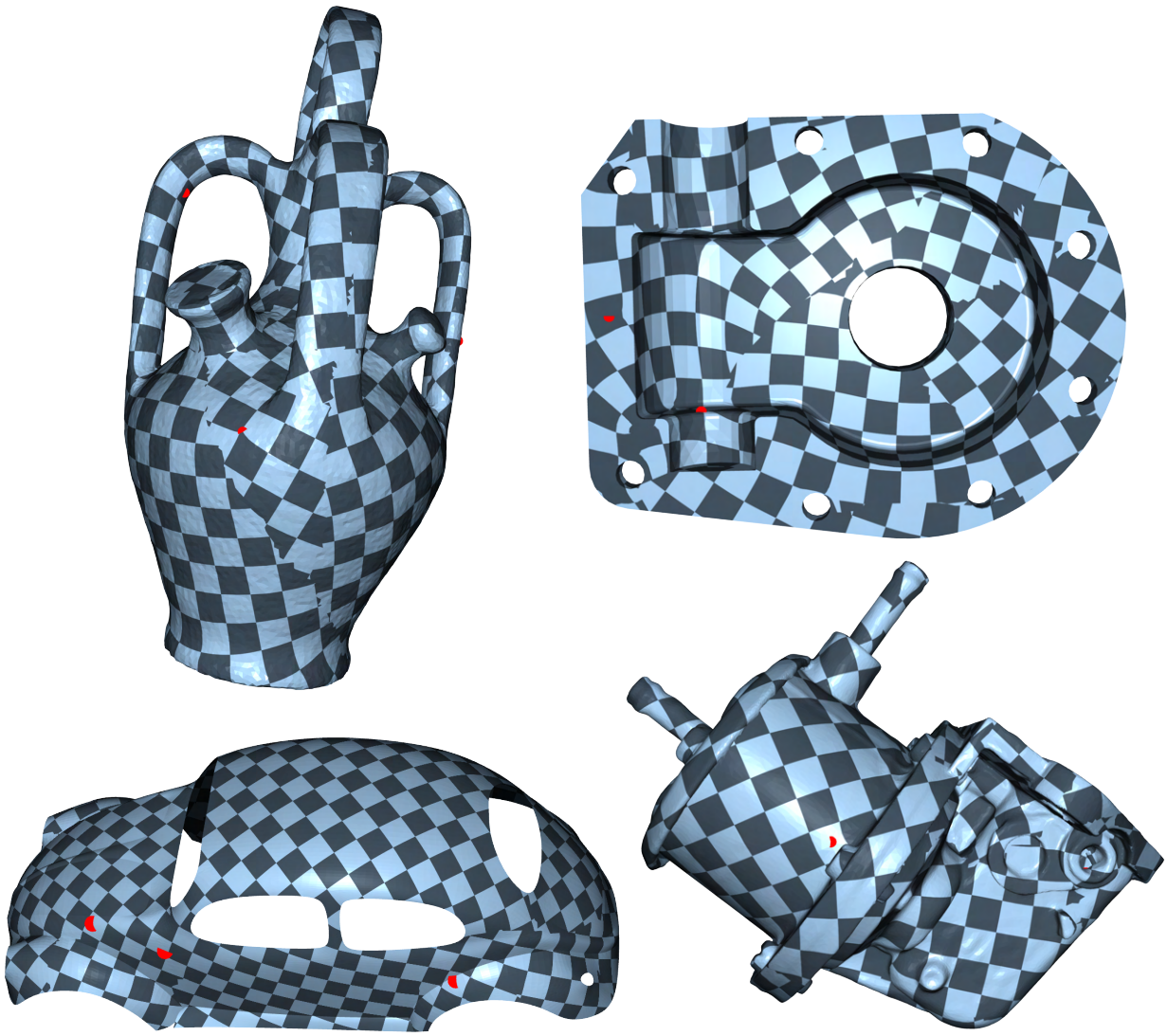


Figure 3.10: A selection of locally injective parametrizations computed by our algorithm by fixing 3 random points to 3 random points in UV space.

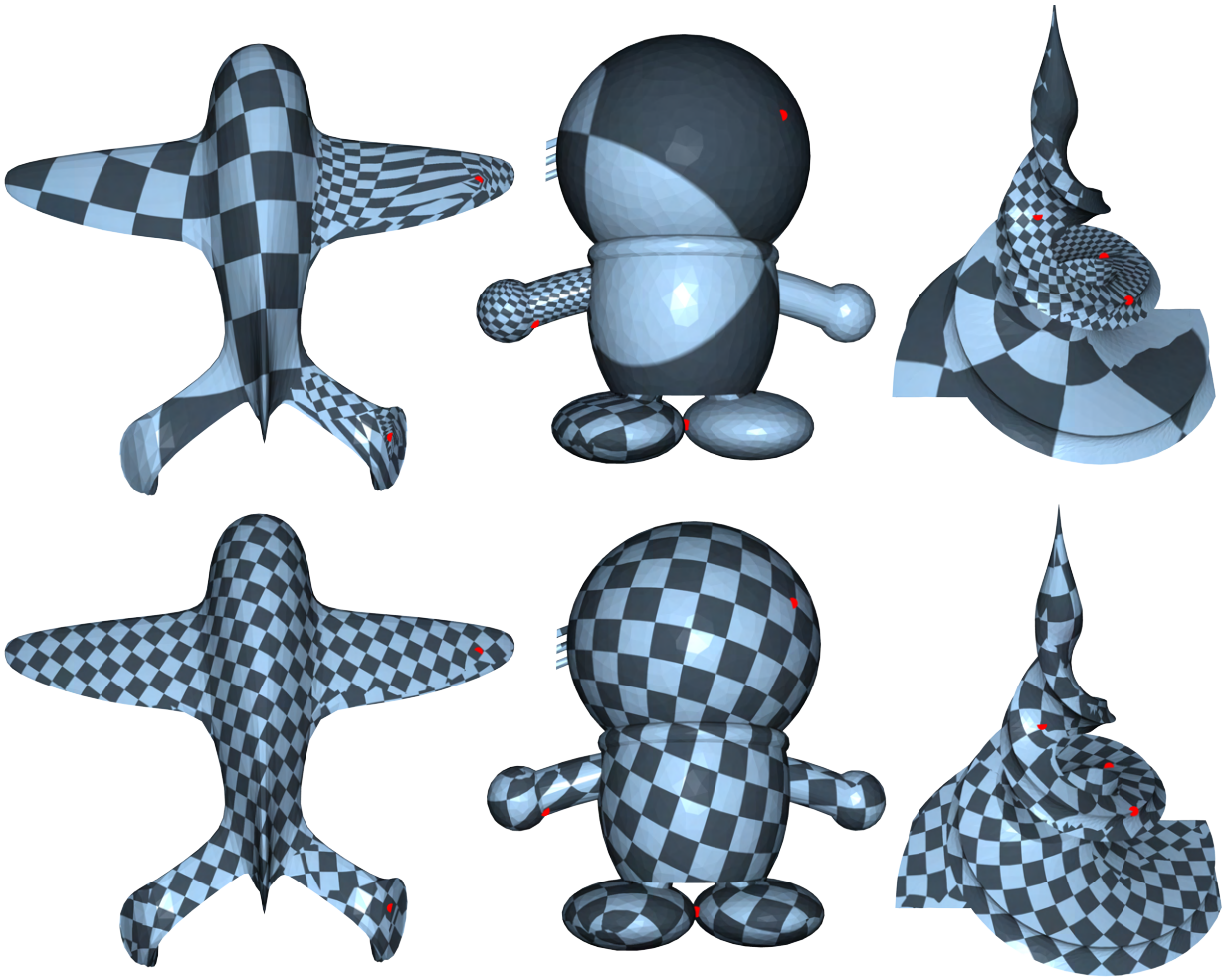


Figure 3.11: Our parametrizations (bottom) have no flipped elements and have a higher quality than those generated by [Kovalsky et al. 2015] (top) using the same positional constraints.

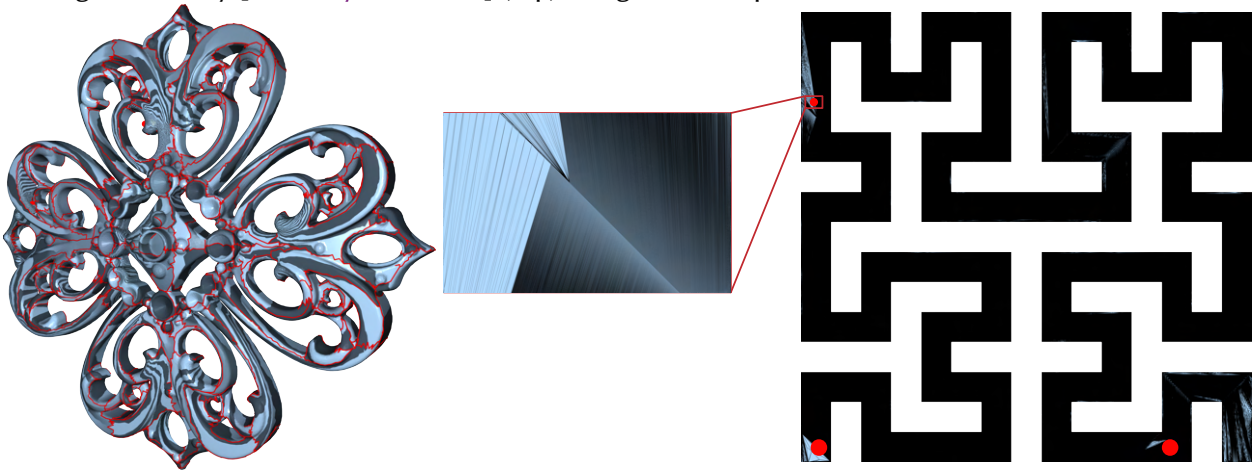


Figure 3.12: To stress test the robustness of MatchMaker++, we parametrize complex surface meshes inside a space filling curve, with 3 additional random positional constraints in its interior.

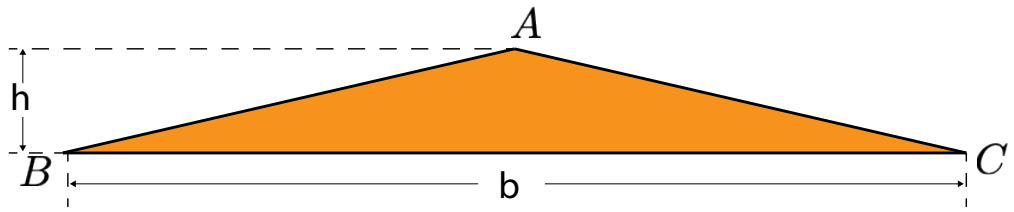


Figure 3.13: A failure case of our implementation in double precision floating point: a triangle without possible points inside. A, B, and C has coordinates $(0, 1 + h)$, $(-b/2, 1)$, and $(b/2, 1)$ resp., where $h = 2^{-53}$(The illustration is not to scale.)

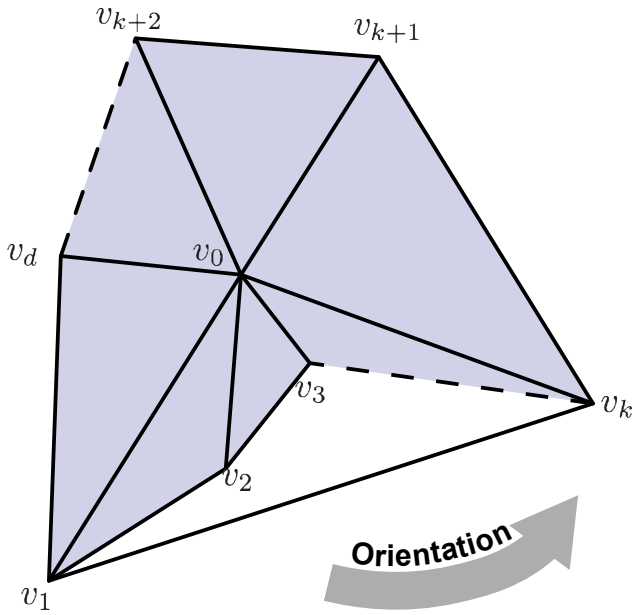


Figure 3.14: Neighbors of v_0 as described as in Lemma 3.2

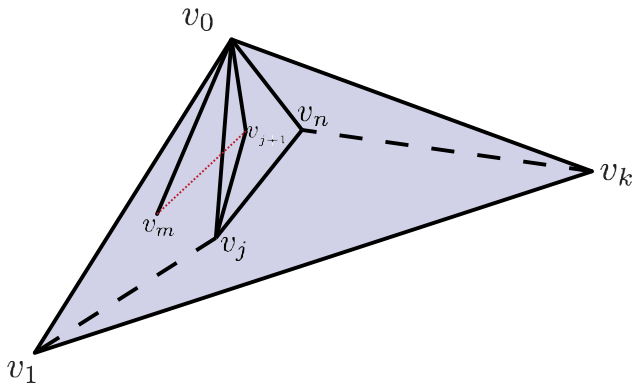


Figure 3.15: Neighbors of v_0 as described as in the proof Lemma 3.3, notice that v_{j+1} is enclosed in $\Delta v_0 v_j v_n$, so a connection to a previous vertex (red dotted line) is forbidden.

4 Efficient and Robust Discrete Conformal Equivalence with Boundary

4.1 Introduction

Computing discrete metrics with prescribed angles on meshes is a problem closely related to surface parametrization and quadrangulation, which is of interest in many geometric settings. Despite many years of efforts, only a few techniques for mesh parametrization provide theoretical guarantees, commonly derived from the same source: discrete harmonic mappings with convex boundary, based on Tutte's embedding theorem [Floater 1997b].

Recent exciting advances concerning the theory of discrete metric uniformization [Gu et al. 2018b; Springborn 2019] provide a solid foundation for a much needed addition to this spectrum of methods. They enable the computation of discrete metrics with arbitrary prescribed discrete curvature at vertices, as long as the discrete Gauss-Bonnet theorem is respected. In particular, this allows to compute, with guarantees, flat metrics or almost-everywhere flat cone metrics with prescribed curvatures at cones—an essential component of global parametrization and quadrangulation algorithms. Guarantees follow from a reduction to an unconstrained convex optimization problem. However, compared to Tutte's method, the numerics involved are far more complex, in

particular due to nonlinearity and large scale distortions inherent in conformal maps.

We present an efficient numerical algorithm based on these new theoretical ideas, extend it to support surfaces with boundary, and explore its practical performance, focusing on robustness.

Problem Summary. To define the problem more precisely, consider a manifold triangle mesh M, possibly with boundary. For a given discrete metric on M, i.e., an assignment of lengths to its edges that satisfy the triangle inequality, we can compute inner angles of triangles.

Let Θ_i be the total angle (the sum of incident inner angles) at vertex v_i, and κ_i its angle deficit, defined as $2\pi - \Theta_i$ for interior vertices and $\pi - \Theta_i$ for boundary vertices. This quantity κ_i can be viewed as the discrete Gaussian curvature if v_i is an interior vertex and the geodesic curvature of the boundary if v_i is on the boundary. Given *target* curvatures $\hat{\kappa}_i$ (respecting the discrete Gauss-Bonnet theorem) our goal is to compute edge lengths that exhibit exactly these curvatures. Flattenings, i.e., mesh parametrizations over the plane, are a special case corresponding to prescribing $\kappa_i = 0$ in the interior [Ben-Chen et al. 2008]. Seamless maps for quadrilateral remeshing are obtained by prescribing $\hat{\kappa}_i = k_i \frac{\pi}{2}$ with $k_i \in \mathbb{Z}$ [Campen et al. 2019; Myles and Zorin 2012].

Approach. As shown in [Gu et al. 2018b; Springborn 2019], a discrete metric realizing target curvatures $\hat{\kappa}_i$ always exists, *if retriangulation of the surface is allowed.* When restricting to metrics *discretely conformally equivalent* to a given original metric, this metric is unique (up to scale) and can be computed by minimizing a convex function.

While the latter property has been exploited before for practical parametrization purposes [Springborn et al. 2008], the assumption of a fixed triangulation restricts the space of target curvatures that can be realized by a conformally equivalent metric. For example, a vertex v_i of valence k cannot, under any (Euclidean) metric, exhibit a discrete curvature $\kappa_i \leq (2-k)\pi$, because inner angles are bounded by π. As a consequence, the resulting discrete metric's edge lengths violate the triangle inequality in some places. This limitation can be remedied by allowing changes to the triangulation of the input surface.

More concretely, the main requirement for triangulation changes needed to enable this is that at all times the triangulation remains an intrinsic Delaunay triangulation. This leads to a natural algorithm [Sun et al. 2015] in the spirit of kinetic data structures [Basch et al. 1999], which, however, requires the determination of the exact sequence of all individual Delaunay-critical events during the metric computation process.

Contributions. In this paper we describe an efficient and practical algorithm, performing triangulation changes with greater flexibility, enabled by the theoretical connection to hyperbolic metrics established by [Gu et al. 2018b; Springborn 2019]. While this theory is developed for closed surfaces, in practice many, if not most, relevant applications involve surfaces with boundaries. These cases can be reduced to the closed surface case by creating a surface double, but a number of algorithmic issues need to be addressed to reliably maintain symmetric intrinsic Delaunay triangulations in such cases. We introduce a number of additional improvements to the basic algorithm, to speed up convergence and increase accuracy and robustness. We furthermore perform extensive evaluations, with a focus on numerical aspects such as the effect of varying arithmetic accuracy. Numerical behavior of the algorithm is of critical relevance as conformal metrics and maps can unavoidably exhibit very large ranges of scales.

We discuss the relevant background in 4.2. An implementation of the main ideas, with particular attention to practical aspects is described in 4.3. Generalization to surfaces with boundary is presented in 4.4, followed by the construction of a surface mapping from the discrete metric in 4.5, and concluded by the evaluation of the algorithm in 4.6.

4.2 Background

We begin by considering the case of surfaces *without* boundary, i.e., we are given a closed manifold triangle mesh $M = (V, E, F)$. The case of surfaces with boundary can be reduced to the closed

surface case with an additional symmetry structure, which we address in detail in 4.4.

The mesh M is equipped with an input metric defined by edge lengths $\boldsymbol{\ell} : E \to \mathbb{R}^{>0}$, satisfying the triangle inequality.

4.2.1 Conformal Equivalence

A *conformally equivalent* discrete metric, for a fixed triangulation M, is defined by means of *logarithmic scale factors* $\boldsymbol{u} : V \to \mathbb{R}$ associated with vertices $V = (v_1, \ldots, v_n)$, by defining new edge lengths as

$$\ell_{ij}(\boldsymbol{u}) = \ell_{ij}\, e^{\frac{u_i + u_j}{2}} \tag{4.1}$$

per edge e_{ij} [Luo 2004]. Given per-vertex target angles $\hat{\Theta}_i$ a conformally equivalent metric with these angles is characterized by, for all i:

$$g_i(\boldsymbol{u}) := \hat{\Theta}_i - \Theta_i(\boldsymbol{u}) = \hat{\Theta}_i - \sum_{T_{ijk}} \alpha^i_{jk}(\boldsymbol{u}) = 0, \tag{4.2}$$

where the inner angle $\alpha^i_{jk}(\boldsymbol{u})$ is computed under the metric defined by $\boldsymbol{u}$ via 4.1, i.e., from edge lengths $\tilde{\boldsymbol{\ell}} = \boldsymbol{\ell}(\boldsymbol{u})$. We use shorthands $\tilde{\boldsymbol{\alpha}} = \boldsymbol{\alpha}(\boldsymbol{u})$ and $\tilde{\boldsymbol{\ell}} = \boldsymbol{\ell}(\boldsymbol{u})$ for these scale factor dependent quantities in the following.

It is known that $\boldsymbol{g}(\boldsymbol{u}) = (g_1(\boldsymbol{u}), \ldots, g_n(\boldsymbol{u}))$ is the gradient of a twice-differentiable convex function [Springborn et al. 2008]. Hence, one may obtain factors $\boldsymbol{u}$ satisfying 4.2 using (second-order) convex optimization methods, starting from arbitrary initializations (e.g. $\boldsymbol{u} \equiv \boldsymbol{0}$). This is true, however, only as long as there is a solution for which $\boldsymbol{u}$ stays in the feasible region $\Omega_M \subset \mathbb{R}^n$ where $\boldsymbol{\ell}(\boldsymbol{u})$ respects the triangle inequality for each triangle T_{ijk}; otherwise it does not define a Euclidean surface metric on M.

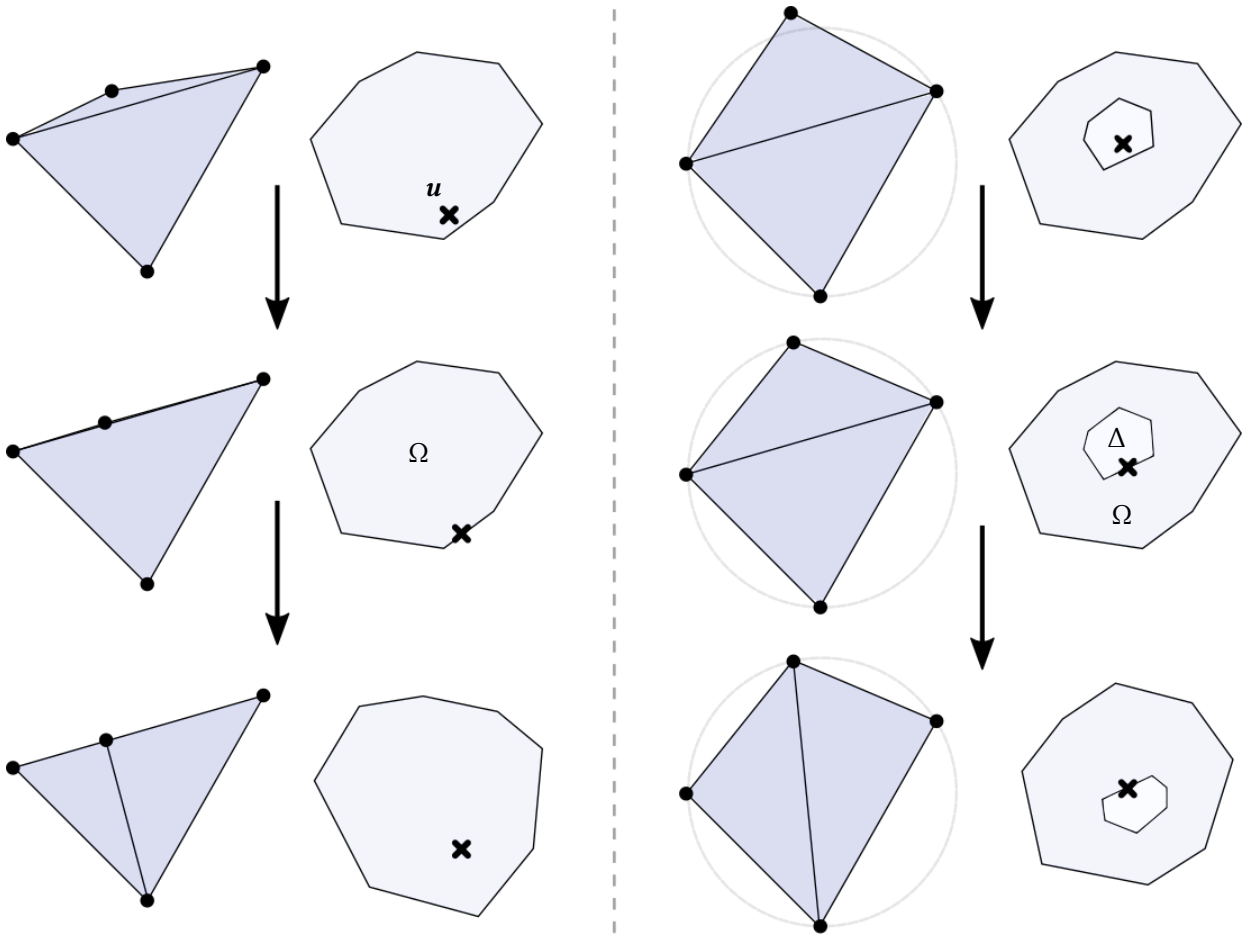


Figure 4.1: Left: flip-on-degeneration. Right: flip-on-Delaunay-violation. Alongside a conceptual illustration of the valid region Ω (light blue) and Delaunay region Δ (white) is shown (cf. section 4.2.2), containing the current point $\boldsymbol{u}$ (cross mark) and changing due to the flip.

4.2.2 Dynamic Triangulation

The feasible region Ω_M can be altered by adjusting the triangulation dynamically *during* the evolution of $\boldsymbol{u}$ from $\mathbf{0}$ towards $\boldsymbol{u}^*$.

Note that a change of triangulation is possible without *intrinsically* changing the surface. M together with given edge lengths defines a surface S_V with a metric which is flat everywhere except at V. There are many triangulations (besides M) with vertices V and their own associated edge lengths, defining the same surface S_V (cf. [Sharp et al. 2019]); hence the differentiation between M and S_V. In particular, an edge flip replacing a pair of triangles (T_{ijk}, T_{jim}) sharing

an edge e_{ij}, with triangles (T_{kim}, T_{mjk}) sharing edge e_{km} can be performed without intrinsically changing the surface S_V, by setting the length of the new edge e_{km} to the length of the diagonal of the planar quadrilateral obtained by unfolding T_{ijk}, T_{jim} [Fisher et al. 2007]. This is referred to as *intrinsic flip*.

Delaunay Flips [Gu et al. 2018b; Springborn 2019] prove a remarkable fact: the convex energy can be extended to all of the space $\mathbb{R}^n$ of scale factors $\boldsymbol{u}$ defined at vertices, *if a particular change of triangulation is allowed*. Specifically, the triangulation is modified so that it stays (intrinsically) Delaunay at all times as $\boldsymbol{u}$ evolves. More specifically, whenever the Delaunay condition is violated as a result of a change in $\boldsymbol{u}$, a flip is performed to maintain the Delaunay property. As the resulting energy is a globally convex function, it can be minimized by an unconstrained Newton method, and the resulting choice of $\boldsymbol{u}$ satisfies (4.2) with respect to the resulting triangulation.

Definition 4.1 (Intrinsic Delaunay)**.** A triangulation is intrinsically Delaunay if the angles of two triangles T_{ijk} and T_{jim} opposite a common edge e_{ij} satisfy the Delaunay condition:

$$\tilde{\alpha}_{ij}^k + \tilde{\alpha}_{ij}^m \leq \pi \tag{4.3}$$

or equivalently $\cos \tilde{\alpha}_{ij}^k + \cos \tilde{\alpha}_{ij}^m \geq 0$. Expressed directly in terms of edge lengths this condition is equivalent to

$$\frac{\tilde{\ell}_{jk}^2 + \tilde{\ell}_{ki}^2 - \tilde{\ell}_{ij}^2}{\tilde{\ell}_{jk}\tilde{\ell}_{ki}} + \frac{\tilde{\ell}_{jm}^2 + \tilde{\ell}_{mi}^2 - \tilde{\ell}_{ij}^2}{\tilde{\ell}_{jm}\tilde{\ell}_{mi}} \geq 0. \tag{4.4}$$

This latter version of the Delaunay condition is particularly important for our construction.

Generically (iff these weak inequalities hold strictly), the intrinsic Delaunay triangulation is unique, but for special configurations (four or more intrinsically co-circular vertices resulting in equality in 4.4) it is not.

For a given triangulation M, the *Penner cell* $\Delta_M \subset \mathbb{R}^n$ denotes the set of scale factors $\boldsymbol{u}$ for which M, along with the modified metric defined by $\boldsymbol{u}$, is intrinsic Delaunay. Clearly, $\Delta_M \subset \Omega_M$,

and when $\boldsymbol{u} \in \partial\Delta_M$ the Delaunay triangulation is not unique. Whenever $\boldsymbol{u}$ reaches the boundary of Δ_M, we can switch to another Delaunay triangulation M' by means of an intrinsic flip, thereby changing the region (from Δ_M to $\Delta_{M'}$), enabling $\boldsymbol{u}$ to evolve further, see 4.1. The cells Δ_M form a partition of $\mathbb{R}^n$ [Gu et al. 2018b].

Such changes of scale factors together with intrinsic Delaunay flips lead to the following generalized notion of discrete conformal equivalence of two metrics [Gu et al. 2018b]:

Definition 4.2 (Discrete Conformal Equivalence)**.** Two metrics $(M_1, \boldsymbol{\ell}_1)$ and $(M_m, \boldsymbol{\ell}_m)$ are *discretely conformally equivalent*, if there is a sequence of meshes with the same vertex set, $(M_s, \boldsymbol{\ell}_s)$, $s = 1, \ldots, m$, such that, for each s, M_s is an intrinsic Delaunay triangulation for the metric $\boldsymbol{\ell}_s$ and either

- $(M_s, \boldsymbol{\ell}_s)$ and $(M_{s+1}, \boldsymbol{\ell}_{s+1})$ are different metrics with the same triangulation (i.e., $M_s = M_{s+1}$) and the edge lengths are related by 4.1 for a choice of $\boldsymbol{u}_s : V \to \mathbb{R}$.
- $(M_s, \boldsymbol{\ell}_s)$ and $(M_{s+1}, \boldsymbol{\ell}_{s+1})$ are different Delaunay triangulations for the same metric.

Degeneration Flips The alternative of performing a triangulation change only when $\boldsymbol{u}$ reaches the boundary $\partial\Omega$ of the currently feasible region was considered by [Luo 2004]. This occurs when a triangle becomes a degenerate cap. An intrinsic flip of this triangle's longest edge yields a non-degenerate triangulation, effectively changing the valid region Ω such that $\boldsymbol{u}$ lies strictly in its interior. 4.1 left illustrates this. An implementation of this approach is described and applied in [Campen and Zorin 2017b]. Open theoretical questions remain regarding the finiteness of the flip sequence and the sound handling of simultaneous adjacent degeneracies.

At first sight, the approach based on maintaining an intrinsic Delaunay triangulation may seem less efficient in comparison. Due to $\Delta \subset \Omega$, at least as many, but often many more cells Δ need to be traversed. Practically, this suggests a large number of small steps between flips in the

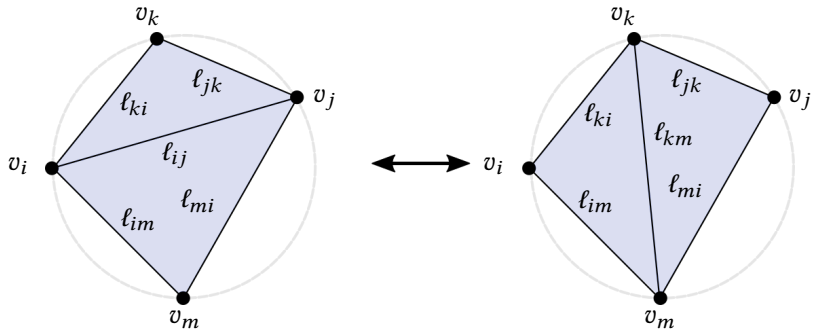


Figure 4.2: Ptolemy flip of an edge e_{ij} shared by two triangles forming an inscribed quadrilateral, i.e., a Delaunay-critical edge.

process of optimizing $\boldsymbol{u}$, compared to, e.g., the use of (less frequent) degeneration flips, and much smaller steps compared to typical unconstrained optimization.

Remarkably, however, this Delaunay approach permits an implementation that is in general more efficient and more robust (see 4.6.2 for a comparison). As we will see, exploiting a relation to *hyperbolic* Delaunay triangulation, arbitrarily large steps can be made, beyond Δ and even beyond Ω (unconstrained by Euclidean triangle inequalities). Flips can be performed collectively *after the fact* and in *arbitrary order*. This is detailed in 4.2.4.

4.2.3 Evolution Step

Assume we are given a triangulation M that is intrinsic Delaunay under the metric defined by some $\boldsymbol{u}_{\vdash}$. Consider an evolution of $\boldsymbol{u}$ from $\boldsymbol{u}_{\vdash}$ to $\boldsymbol{u}_{\dashv}$, e.g., linear:

$$\boldsymbol{u}(t) = (1-t)\boldsymbol{u}_{\vdash} + t\boldsymbol{u}_{\dashv}, \;\; t \in [0,1].$$

As we move along the interval $[0,1]$, whenever four vertices forming triangles T_{ijk} and T_{jim} become co-circular under the metric defined by $\boldsymbol{\ell}(\boldsymbol{u}(t))$, an intrinsic flip of edge e_{ij} is performed. Due to the special configuration (the two triangles forming an *inscribed* quadrilateral, see 4.2) the

length that the new edge e_{km} needs to take can be computed following Ptolemy's theorem as

$$\tilde{\ell}_{km} = \frac{1}{\tilde{\ell}_{ij}}(\tilde{\ell}_{jk}\tilde{\ell}_{im} + \tilde{\ell}_{ki}\tilde{\ell}_{mj}), \tag{4.5}$$

where we use $\tilde{\boldsymbol{\ell}}$ as a shorthand for $\boldsymbol{\ell}(\boldsymbol{u}(t))$. For $\tilde{\ell}_{km} = \ell_{km}(u(t)) = \ell_{km}\, e^{\frac{u_k+u_m}{2}}$ to take this value for the current $\boldsymbol{u}(t)$, we need to set:

$$\ell_{km} := \frac{1}{\ell_{ij}}(\ell_{jk}\ell_{im} + \ell_{ki}\ell_{mj}). \tag{4.6}$$

Notice that this is Ptolemy's formula, 4.5, applied to the *original* metric, as all scale factors cancel. In other words: applying the formula in the current ($\boldsymbol{u}(t)$-scaled) metric $\tilde{\boldsymbol{\ell}}$ is equivalent to applying it in the original metric $\boldsymbol{\ell}$, followed by scaling. Remarkably, this holds *even though the vertices are not co-circular under the original metric* in general. Moreover, the edge lengths $\boldsymbol{\ell}$ set in this way may *not even satisfy the triangle inequality*. This is no issue, though, as certainly the relevant scaled lengths $\tilde{\boldsymbol{\ell}} = \boldsymbol{\ell}(\boldsymbol{u}(t))$ do, by construction.

It was shown that the number of flip events along the path is finite [Wu 2014], which means that after a finite number of flips we will obtain the triangulation and edge length assignment needed for the target $\boldsymbol{u}(1) = \boldsymbol{u}_\dashv$.

One practical downside of this procedure, in which the necessary flips along the evolution path are detected and performed one-by-one sequentially [Sun et al. 2015], is that it requires solving precisely for the sequence of flips. An alternative approach, whose correctness can be shown based on an interpretation of the involved edge lengths as defining hyperbolic metrics instead of Euclidean metrics, improves on this.

4.2.4 Hyperbolic Metric Approach

Instead of moving t along the interval $[0, 1]$, determining the sequence of flip events and executing them in order, let us directly consider $t = 1$. The initial triangulation M may not be Delaunay under $\boldsymbol{u}(1)$, and the edge lengths $\boldsymbol{\ell}(\boldsymbol{u}(1))$ may not even respect the triangle inequality. Nevertheless, we can test each edge for violation of the Delaunay criterion using 4.4 applied to $\boldsymbol{\ell}(\boldsymbol{u}(1))$, and incrementally flip (using 4.6) all violating edges in arbitrary order following the classical flip algorithm until a Delaunay triangulation is reached [Bobenko and Springborn 2007]. While in case of triangle inequality violations this criterion lacks the geometric justification via 4.3 (the involved quantities are no longer (cotangents of) Euclidean angles), this algorithm nevertheless succeeds.

Hyperbolic Delaunay. The reasons for applicability of 4.4 and use of 4.6 are direct consequences of an elegant correspondence between hyperbolic and conformal metric structures used in the proofs of [Gu et al. 2018b; Springborn 2019] and introduced in [Rivin 1994], given by mapping edge lengths to Penner coordinates of a hyperbolic metric, and Euclidean triangulations to ideal triangulations. Detailed explanations can be found in these papers and an overview given in [Crane 2020, §5, §6]. We go into more detail in 4.5, as this relation is important when the conformal metric is used to establish a conformal map, for purposes of evaluation of the map at arbitrary points. Here we just present a proposition summarizing the aspect of this theory relevant to our algorithm.

Proposition 4.3. *Suppose lengths $\tilde{\boldsymbol{\ell}}$ (possibly not satisfying triangle inequality) are assigned to edges in a triangle mesh M, conformally equivalent to a set of Euclidean metric lengths $\boldsymbol{\ell}$. If the flip algorithm is applied to $\tilde{\boldsymbol{\ell}}$, with the Delaunay criterion in algebraic form* (4.4) *used to determine which edges need to be flipped, and the Ptolemy formula* (4.6) *used for length updates, the algorithm produces a triangulation M' with lengths $\tilde{\boldsymbol{\ell}}'$ that satisfy the triangle inequality. This triangulation is*

intrinsic Delaunay. Moreover, the discrete metric defined by $(M', \tilde{\boldsymbol{\ell}}')$ *is discrete conformally equivalent to* $(M, \boldsymbol{\ell})$.

In summary, instead of performing flips following an expensive-to-compute sequence required to maintain a valid Euclidean metric on triangles at all times, the algorithm performs the flips in arbitrary order, yielding edge lengths $\tilde{\boldsymbol{\ell}}$ satisfying the triangle inequality only in the end. This version of the flip algorithm is referred to as *Weeks algorithm* [Weeks 1993].

These observations ensure that whenever we modify scale factors $\boldsymbol{u}$ while computing the conformal metric, the flip algorithm can be used to recover a Delaunay triangulation, which can then be used to evaluate the value of the convex function we need to minimize, its gradient, and its Hessian.

4.3 Algorithm

Using this background, we can now formulate an efficient algorithm for the computation of a conformally equivalent metric, respecting prescribed target angles $\hat{\boldsymbol{\Theta}}$. The algorithm, spelled out in 3, is based on a standard Newton's method with line search, but incorporates several important details and modifications.

Delaunay. Initially, if M is not already intrinsically Delaunay, it is turned into a Delaunay mesh using standard intrinsic edge flips. Then, whenever $\boldsymbol{u}$ is updated (during the line search), before the gradient and Hessian are evaluated the triangulation is turned into an intrinsic Delaunay triangulation with respect to the metric defined by $\boldsymbol{u}$ using Weeks flip algorithm—now using the Ptolemy length computation rule from 4.6.

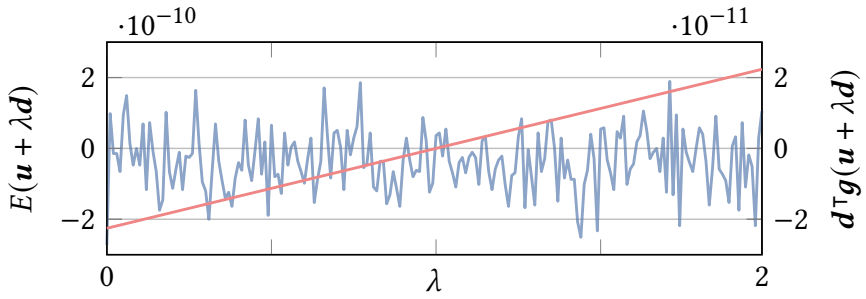


Figure 4.3: Energy (blue; mean (20202.12) subtracted) and projected gradient (red) along a descent direction $\boldsymbol{d}$. Notice that the numerical noise in the energy computation dominates the actual change in energy, making it less suitable to be a measure of progress in the line search. By contrast, the sign of the projected gradient (red) can be determined much more precisely.

Energy-free Line Search. The function $E(\boldsymbol{u})$ that needs to be minimized is known explicitly [Springborn et al. 2008]:

$$E(\boldsymbol{u}) = \sum_{T_{ijk}} \Big(2f(\tilde{\lambda}_{ij}, \tilde{\lambda}_{jk}, \tilde{\lambda}_{ki}) - \pi(u_i + u_j + u_k) \Big) + \hat{\boldsymbol{\Theta}}^{\mathsf{T}}\boldsymbol{u},$$

where $\tilde{\lambda}_{ij} = 2\log \ell_{ij} + u_i + u_j$ and f is a per-triangle function involving Milnor's Lobachevsky function [Springborn et al. 2008, Eq. (8)]. The gradient of $E(\boldsymbol{u})$ is $\boldsymbol{g}(\boldsymbol{u}) = \hat{\boldsymbol{\Theta}} - \boldsymbol{\Theta}(\boldsymbol{u})$ (4.2) and its Hessian $H(\boldsymbol{u})$ simply is the well-known positive semi-definite cotan-Laplacian in terms of the scaled angles $\boldsymbol{\alpha}(\boldsymbol{u})$.

The obvious approach is to use the standard Newton's method with backtracking line search, using $E(\boldsymbol{u})$, $\boldsymbol{g}(\boldsymbol{u})$, $H(\boldsymbol{u})$ (cf. [Gillespie et al. 2021]). However, computing the energy $E(\boldsymbol{u})$, in particular evaluating the Lobachevsky function, presents numerical challenges, and efficient Chebyshevpolynomial approximations, like the one used in the implementation of [Springborn et al. 2008], may not yield sufficient accuracy, while incurring additional computational overhead. We observe that the energy can be very flat along the search direction, so using the decrease of energy evaluated this way as a criterion in the line search may lead to the algorithm stalling due to numerical noise (see 4.3). This is particularly problematic in cases requiring high conformal distortion or if we want to compute the conformal metric with high precision, as needed for instance

to derive an implied conformal map (cf. 4.6).

The evaluation of the gradient and Hessian, both of which are simple functions of angles (not involving the Lobachevsky function), by contrast, is more efficient and numerically robust than the energy itself (see 4.3). Fortunately, we are able to formulate our algorithm such that it relies on $\boldsymbol{g}(\boldsymbol{u})$ and $H(\boldsymbol{u})$ only. This is possible for the following reason: As $E(\boldsymbol{u})$ is convex, it is also convex along the search direction $\boldsymbol{d}$, i.e., $E(\boldsymbol{u} + \lambda\boldsymbol{d})$ for fixed $\boldsymbol{u}$ and $\boldsymbol{d}$ is convex in the step size λ. Therefore its derivative

$$\frac{\partial}{\partial\lambda}E(\boldsymbol{u} + \lambda\boldsymbol{d}) = \boldsymbol{d}^{\mathsf{T}}\boldsymbol{g}(\boldsymbol{u} + \lambda\boldsymbol{d}), \tag{4.7}$$

i.e., the gradient's projection onto the search direction, has at most one zero. Hence, if we require that the step size λ is selected in the line search such that $\boldsymbol{d}^{\mathsf{T}}\boldsymbol{g}(\boldsymbol{u} + \lambda\boldsymbol{d}) \leq 0$, this guarantees that $E(\boldsymbol{u})$ decreases, without the need for checking the function value itself.

Note that avoiding energy evaluation precludes the use of standard *sufficient decrease conditions* (most commonly, Armijo condition) in the line search. However, a simple backtracking search, starting with $\lambda = 1$, for a point along the search direction with negative projected gradient, ensures that the Newton step, when it is less than one, is always in the range $[\lambda_m/2, \lambda_m]$, where λ_m is the function's (unknown) minimum point along the search line. One can show that this is sufficient for convergence by following the standard analysis of Newton's method with inexact line search. However, this is, in general, not sufficient to guarantee that the algorithm converges *quadratically*. An additional Armijo-like condition (the first termination condition in the line search in 3; we use $\alpha = 0.1$, with a meaning similar to the Armijo condition constant) yields a more consistent quadratic behavior. The practical effect of this additional termination condition is small in most cases (most commonly, the reduction in the number of iterations on our test datasets is around 1-2). A detailed analysis of convergence of the proposed energy-free method can be found in [Zorin 2021].

Termination. The accuracy with which the target angles $\hat{\boldsymbol{\Theta}}$ can be matched of course depends (in a non-trivial manner) on the precision of the real number representation. If tolerance ε_{tol} is chosen too low relative to this, 3 may never terminate. For practical purposes therefore additional stopping criteria can be taken into account: an upper bound on the number of Newton steps and the number of line search halvings, a lower bound on the Newton decrement $\boldsymbol{d}^\intercal \boldsymbol{g}(M, \boldsymbol{\ell}, \boldsymbol{u})$. Information about the practically achievable accuracy can be found in 4.6.3.

Additional Performance Heuristic. In particularly challenging cases, the gradient direction and in particular its magnitude can be rapidly varying. The line search loop may then have to be executed many times before a valid step size is found, causing many redundant edge flips. One additional line search heuristic that proved beneficial in this regard is a *gradient norm decrease* condition. Specifically, as a stopping condition for the line search we require that, in addition to $\boldsymbol{d}^\intercal \boldsymbol{g} < 0$, the norm of the gradient $\|\boldsymbol{g}\|$ decreases. Only if this additional condition forces the step size below a given threshold (we use 10^{-10}), the condition is lifted for one step, allowing the gradient to grow, so as to not hamper convergence.

Overlay Mesh. An embedding of the (by edge flips) modified mesh in the original mesh can be maintained by using a mesh overlay data structure. Towards the algorithm it behaves like a mesh, but internally it keeps track of the *overlay* of both meshes, updating it whenever an edge is flipped. [Fisher et al. 2007] propose to represent it explicitly by means of a polygon mesh data structure, [Gillespie et al. 2021] propose a more lightweight implicit representation by normal coordinates. We found the overhead of even the explicit structure to be benign (e.g., on average 11% added time cost on the example cases from 4.9).

4.4 Boundaries

So far, we assumed that M is a closed surface. For a surface with boundary, the problem can be reduced to the case of closed surfaces by gluing a mirrored copy to the surface along the boundary, turning it into a closed surface with an obvious (reflectional) symmetry. A strategy of this kind is also used in [Sun et al. 2015] and [Gillespie et al. 2021].

However, the initial symmetry of the setting may be disturbed when applying 3. Due to numerical inaccuracies, the values $\boldsymbol{u}$ may diverge on the two copies; application of a standard Delaunay flip algorithm is further complicated by the presence of stably cocircular configurations, as we discuss below. Therefore we describe a version of this surface double cover approach that explicitly imposes and maintains symmetry, on the numerical as well as the combinatorial level, by construction.

4.4.1 Double Cover

Let the input surface be N. Its double cover is constructed as follows:

1. we attach a mirrored copy N' of the input mesh N along the boundary (merging boundary vertices and edges), as illustrated below, yielding a closed mesh M,
2. we transfer the edge lengths $\boldsymbol{\ell}$ and the target curvatures κ_i of interior vertices v_i from N to N',
3. we prescribe $\hat{\Theta}_i = 2\pi - 2\hat{\kappa}_i$ at each (former) boundary vertex v_i, where $\hat{\kappa}_i$ is the target discrete geodesic boundary curvature at vertex v_i.

The double cover mesh M built this way exhibits an obvious reflectional symmetry, i.e., there is a map R with $R^2 = I$ that takes vertices to vertices, edges to edges, and faces to faces. It maps an element in the interior of N to its copy in N' and vice versa; on the merged (former) boundary vertices and edges, R is the identity.

Conformal Metric Symmetry. Due to symmetry (i.e., invariance with respect to R) of the mesh M, the metric $\boldsymbol{\ell}$, and the target angles $\hat{\boldsymbol{\Theta}}$, the symmetrically initialized factors $\boldsymbol{u}$ will (in theory, up to numerical round-off error) remain symmetric after each iteration of the optimization process. This can be seen by observing that the function $E(\boldsymbol{u})$ is the sum of per-triangle terms $E_T(\boldsymbol{u}_T)$, where $\boldsymbol{u}_T$ is the restriction of $\boldsymbol{u}$ to vertices of the triangle T. Given the above symmetry, its gradient $\boldsymbol{g}(\boldsymbol{u}) = \nabla_{\boldsymbol{u}} E$ therefore is invariant with respect to R. Consequently, if we cut the mesh along the symmetry line in the end, so as to discard one copy, a boundary vertex v_i will have exactly half the prescribed angle, $\frac{1}{2}\hat{\Theta}_i = \pi - \hat{\kappa}_i$, and therefore exhibit a discrete geodesic boundary curvature of $\hat{\kappa}_i$, just as intended.

Tufted Double Cover. The fact that $\boldsymbol{u}$ (and thus all vertex-associated attributes) are supposed to evolve symmetrically implies that we can use a *tufted* double cover as in [Sharp and Crane 2020], with the unknown scale factors $\boldsymbol{u}$ shared between the two symmetric halves of M, to reduce the number of variables (and to impose perfect symmetry on the numerical level). This does not mean, however, that computations could trivially be restricted entirely to one half of the double cover only: edge flips may, and commonly will, create edges and faces spanning both halves of the double cover, crossing the symmetry line.

Combinatorial Symmetry. Edge flips across the symmetry line can lead to triangulations that are no longer combinatorially symmetric, as depicted in fig. 4.4. Unless special care is taken, this can increase the chance of numerical inaccuracies causing divergence from geometric symmetry. Furthermore, such cases contain co-circular vertex configurations that are *stable*, i.e., for the given triangulation, due to the symmetry of $\boldsymbol{u}$, these remain co-circular *independent of the evolution of* $\boldsymbol{u}$. An example is the diagonal edge on the right in the inset. As in this case, numerical evaluation of the Delaunay condition results in an essentially random choice of the result, in order to avoid potentially infinite flip sequences of Delaunay-critical edges, we instead perform special flips at the symmetry line, maintaining perfect combinatorial symmetry by construction, as detailed

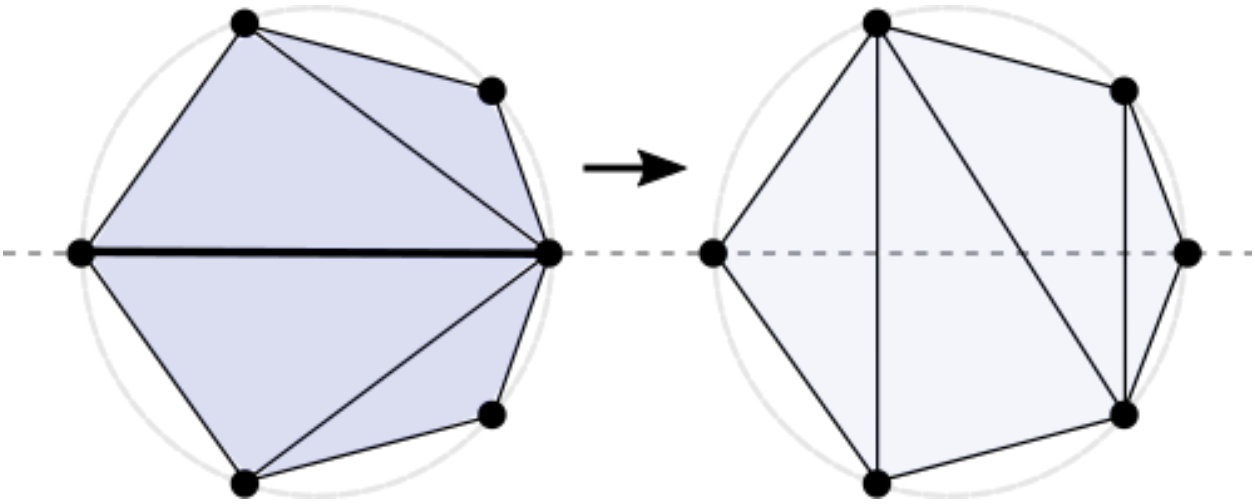


Figure 4.4: Edge flips across the symmetry line can lead to triangulations that are no longer combinatorially symmetric.

in the next section. Our method explicitly identifies these stably cocircular configurations and ensures that Delaunay flips are never applied to these, even if they appear to be slightly non-Delaunay due to numerical inaccuracies. In addition, having a symmetric Delaunay mesh for the final configuration can simplify the extraction of the resulting metric or map for the original surface with boundary.

4.4.2 Symmetric Meshes

Our goal is to rigorously determine which edge flip cases can occur in a symmetric mesh, in particular at the symmetry line, so as to ensure all special cases are correctly handled in our method. To that end, we begin by making precise the general notion of *combinatorially symmetric* polygon mesh. In this, rather than using edges, we use *halfedges*, each associated with a unique face (or a boundary loop, which can be treated exactly like a face). Specifically, each edge corresponds to two halfedges.

Definition 4.4 (Combinatorial Mesh)**.** A combinatorial polygon mesh is a triple $(H, \mathcal{N}, O)$ of a set of halfedges H, a bijective function $\mathcal{N} : H \to N$ (*next-halfedge* function), and a bijective

function O (*opposite-halfedge* function) with the property

$$O^2(h) = h;\ O(h) \neq h \tag{4.8}$$

i.e., all orbits of O have size 2.

This definition is quite general which is important for maintaining intrinsic Delaunay triangulations: e.g., it allows for vertices of valence 1, polygons glued to themselves, etc., all of which are possible configurations in these triangulations.

Definition 4.5 (Mesh Elements)**.** Define the bijective circulator function $C : H \to H$ as $\mathcal{N}^{-1}(O(h))$. Then the mesh has the following implied elements:

- *Faces* are the orbits of the next-halfedge function $\mathcal{N}$.
- *Vertices* are the orbits of the circulator function C.
- *Edges* are the orbits of the opposite-halfedge function O.

Collectively we refer to them as (mesh) *elements*. A halfedge *belongs* to an element if it is part of the respective orbit.

A *mesh with boundary* is a mesh with a subset of its faces marked as boundary loops. The halfedges of these loops form the set H^{bnd} of *boundary halfedges*.

Definition 4.6 (Reflection Map)**.** A reflection map $R : H \to H$ for a mesh $(H, \mathcal{N}, O)$ is an involution ($R^2 = I$) defined on the set of halfedges: each halfedge is mapped either to itself, or forms a reflection pair with a distinct halfedge. It is required to satisfy the following conditions:

1. preservation of O relation: $O(R(h)) = R(O(h))$,
2. inversion of $\mathcal{N}$ relation: $\mathcal{N}(R(h)) = R(\mathcal{N}^{-1}(h))$,

3. preservation of boundary: $h \in H^{bnd} \Leftrightarrow R(h) \in H^{bnd}$.

Note that conditions (1) and (2) correspond to the properties of continuity and orientation-reversal of continuous reflection maps [Panozzo et al. 2012]. They imply that R maps orbits of $\mathcal{N}$, of $\mathcal{O}$, and, as a consequence, of $\mathcal{C}$, to orbits of these functions, i.e., it is well-defined for faces, edges, and vertices (via $R(x) = x' \Leftrightarrow R(h) \in x'$ for any $h \in x$). Furthermore, because $R^2 = I$, all orbits of R have length 1 or 2, whether it acts on halfedges, faces, edges, or vertices. This implies the following partitioning.

Proposition 1 (Halfedge Sets). *H can be partitioned into disjoint sets H^1, H^2, H^s so that the following conditions are satisfied:*

- $h \in H^s \Leftrightarrow R(h) = h$;
- $h \in H^1 \Leftrightarrow R(h) \in H^2$;
- *for any face or edge x, either all belonging halfedges are in H^1, or all in H^2, or x is fixed by R (i.e. $R(x) = x$).*

This leads to the following partitioning of the sets of edges and faces, where $e = (h, h')$, $f = (h_0, \dots h_{m-1})$ denote the orbits of belonging halfedges:

- $e \in E_i \Leftrightarrow h, h' \in H^i$, $i = 1, 2$
- $e \in E^{\perp} \Leftrightarrow h, h' \in H^s$
- $e \in E^{\parallel} \Leftrightarrow h = R(h')$
- $f \in F_i \Leftrightarrow h_0 \in H^i$, $i = 1, 2$
- $f \in F^s \Leftrightarrow R(h_0) \in f$

The set $E^{\perp}$ is the set of edges (perpendicularly) crossing the symmetry line between two halves of a symmetric mesh mapped to each other (see 4.5 right); the set $E^{\parallel}$ is the set of edges on the symmetry line; F^s is the set of faces that cross the symmetry line, and are mapped by the symmetry map to themselves. For additional details, see 4.7.

Using this terminology, our double cover construction from 4.4.1 can be described formally in terms of combinatorial structure of the mesh (see 4.8). Initially we have $E^{\perp} = \emptyset$ and $F^s = \emptyset$, i.e., no element crosses the symmetry line (the former boundary). $E^{\parallel}$ contains the edges lying *on* the symmetry line, i.e., those for whose halfedges the O relation was adjusted to glue the two copies. This initially simple situation can change, however, when edge flips are performed on the double cover mesh.

4.4.3 Symmetric Flips

When an edge e in a symmetric mesh $M = (H, N, O, R)$ shall be flipped, the edge $R(e)$ needs to be flipped as well (unless $R(e) = e$), so as to be able to maintain a symmetric mesh. The simultaneous flip of e and $R(e)$ (as well as the single flip of e if $R(e) = e$) is referred to as *symmetric flip*. As discussed in 4.4.1, in the algorithm from 4.3 the metric evolves symmetrically. This implies that whenever the algorithm intends to flip an edge e, it simultaneously intends to flip $R(e)$ as well. The algorithm is therefore compatible with the restriction to symmetric flips.

While for an edge $e \in E_i$ with incident faces $f, g \in F_i$ the process is obvious, special care needs to be taken when elements from $E^{\parallel}$, $E^{\perp}$, or F^s are involved. We will exhaustively distinguish different types of symmetric flips based on the membership of the involved edges and faces in these sets.

Flip Types For a triple (f_a, e, f_b) of an edge e with incident faces f_a, f_b, the triple of labels denoting their set memberships, e.g., $(1, \parallel, 2)$, is called flip *type* of the edge e.

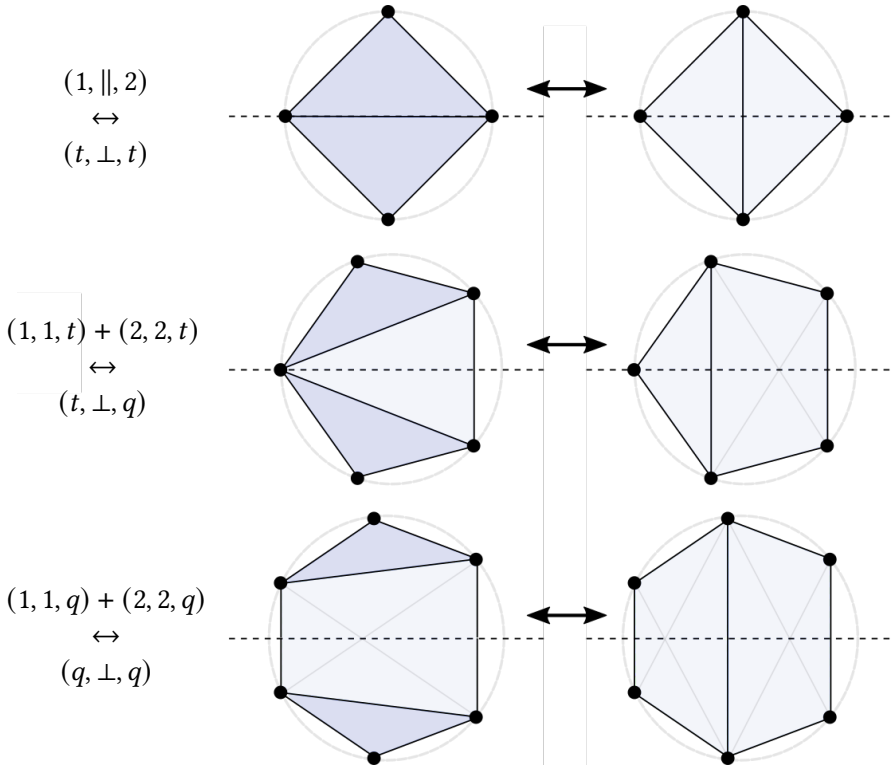


Figure 4.5: Symmetric edge flips involving faces from F^s (light blue), crossing the symmetry line (dashed). Faces from F^1 and F^2 are colored dark blue. The configurations are shown with co-circular vertices, though combinatorially flips can be performed in any state. Note that the light blue quads' vertices, however, are necessarily co-circular by symmetry, regardless of metric.

Consistent Flip Types We say that an edge has a *consistent* flip type, if this particular triple may occur in a symmetric mesh. For instance, $(1, \perp, 1)$ is not a consistent type, as edges from $E^{\perp}$ necessarily have incident faces from F^s by definition.

Proposition 2 (4.7) helps to reduce the possible set to the following six possibilities, up to a $1 \leftrightarrow 2$ exchange. It is easy to construct examples proving that all of them are consistent, i.e., may occur in a symmetric mesh:

- Edge in E^1: Set 1a: $(1, 1, 1)$, $(1, 1, s)$ Set 1b: $(s, 1, s)$
- Edge in $E^{\|}$: Set 2a: $(1, \|, 2)$ Set 2b: $(s, \|, s)$
- Edge in $E^{\perp}$: Set 3: $(s, \perp, s)$

Relevant Flip Types Among these types, only four are also *relevant*; Following Proposition 3 (4.7), types of the form $(s, \|, s)$ and $(s, 1, s)$ in the sets 1b and 2b are necessarily associated with edges that satisfy the Delaunay condition 4.4 irrespective of the choice of lengths of edges involved. These are not relevant for the purpose of the algorithm from 4.3, which exclusively flips non-Delaunay edges. This leaves only sets 1a, 2a, and 3 for further consideration.

Triangles and Quadrilaterals A flip of type $(1, 1, s)$ leads to a pair of triangles in F^s that together form a quadrilateral which is inscribed, i.e., the four vertices are intrinsically co-circular (4.5 center). Remarkably, this statement holds regardless of metric, as long as it is symmetric, i.e., invariant with respect to R. Instead of randomly choosing a diagonal splitting this quadrilateral into two triangles, we explicitly represent it as a quadrilateral face. This avoids violating the symmetry by the diagonal, which, e.g., would complicate recovering the surface with boundary after the conformal metric is computed, and avoids potential issues such as sequences of flips caused by numerically nearly co-circular points.

Faces in F^s can therefore be triangular or quadrilateral. We accordingly partition $F^s = F^t \cup F^q$, and based on this distinguish t-versions and q-versions of flip types involving the label s. This yields a total of seven types that are consistent and relevant.

Six of these seven flip types form three pairs of mutually inverse flips, while one is self-inverse. We can thus succinctly summarize :

1. $(1, 1, 1) + (2, 2, 2) \leftrightarrow (1, 1, 1) + (2, 2, 2)$;
2. $(1, \|, 2) \leftrightarrow (t, \perp, t)$;
3. $(1, 1, t) + (2, 2, t) \leftrightarrow (t, \perp, q)$;
4. $(1, 1, q) + (2, 2, q) \leftrightarrow (q, \perp, q)$.

Case (1) is the standard case of flipping a configuration not involving the symmetry line. (2), (3), and (4) are the special cases crossing the symmetry line; they are illustrated in 4.5. 4.1 details the

combinatorial changes to be performed on the symmetric mesh so as to execute these symmetric flips. In terms of implementation, it thus simply comes down to initially labeling the edges and faces of the double cover, updating the labels when flipping edges, and using one of these special case rules whenever a label other than 1 or 2 is involved in a flip.

4.4.4 Symmetric Metric

To be able to apply 3 to such symmetric meshes to compute a symmetric conformal metric, what is left to clarify is how to deal with quadrilateral faces.

Delaunay Criterion For edges with two incident triangles, the Delaunay check needed for the algorithm is standard, via 4.4. If one of the incident faces is a quad, due to symmetry it, regardless of the metric, is an inscribed trapezoid. As a consequence, whichever way we (virtually) split it into triangles we get the same angles opposite any of its edges. Hence, we may perform the Delaunay check assuming arbitrary virtual diagonals in the quads.

Gradient and Hessian For the same reason, the computation of gradient $g(\boldsymbol{u})$ and Hessian $H(\boldsymbol{u})$ can be performed based on arbitrary diagonals; the choice does not affect the result [Springborn 2019].

Ptolemy Formula Note that each of the edges created by symmetric flips involving quads (4.5) can also be obtained by a sequence of edge flips involving triangles (and split quads) only. In this way the length of such edges can be computed using (multiple instances of) the standard Ptolemy formula 4.6. As there are only four types of flips involving quads, one can conveniently derive closed form expressions for these cases in advance, rather than actually performing these sequences for each flip. Note that each quad needs to store its diagonal length to enable these computations.

4.4.5 Restriction to Single Cover

Once 3 has terminated and the desired conformal metric has been computed, we finally need to discard half of the double cover: we need to cut the symmetric surface along the line of symmetry. While initially the entire symmetry line is formed by a sequence of mesh edges, this may no longer be the case due to flips (unless an overlay is used), namely whenever F^s and $E^\perp$ are not empty in the end. One simply needs to split all edges from $E^\perp$ at their midpoint, and split the triangles and quads from F^s by connecting these inserted split vertices.

4.5 Continuous maps from discrete metrics

The algorithms described in previous sections deal exclusively with discrete metric definitions, i.e., assignments of edge lengths to edges of a mesh. If mesh connectivity does not change, an affine map from the initial mesh triangles T to the final mesh triangles $\tilde{T}$ can be easily inferred from the lengths. However, as pointed out in [Springborn et al. 2008], a natural map is actually a projective map between triangles, which, in addition to mapping the original lengths to conformally deformed ones, also maps the circumcircle of T to the circumcircle of $\tilde{T}$. While for fixed connectivity this yields only a moderate improvement in, e.g., texture quality, for changing connectivity the map definition is more relevant.

While for the discrete algorithm itself we only needed a simple-to-formulate (although surprising) fact that Weeks flip algorithm can be used to obtain an intrinsically Delaunay mesh even if the triangle inequality is violated at intermediate steps, defining maps between the original mesh points and the final (e.g. flat) mesh points requires a more in-depth exposition of the underlying theory.

Our goal in this section is to define a map $f : |M| \to |M'|$, from the original to the final (e.g. conformally flattened) mesh, more specifically, mapping formulas of the form $(w'_l, w'_m, w'_m; T') =$

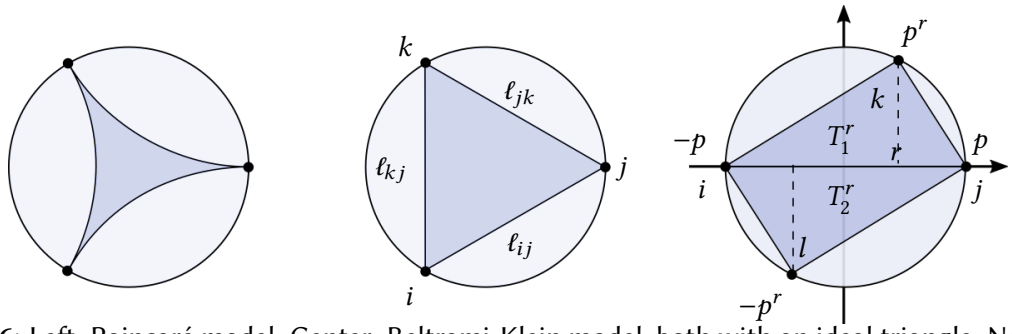


Figure 4.6: Left: Poincaré model. Center: Beltrami-Klein model, both with an ideal triangle. Note that in the Beltrami-Klein model it forms a Euclidean triangle. Right: Two-triangle chart.

$f(w_i, w_j, w_k; T)$, where (w_i, w_j, w_k) are barycentric coordinates of a point on the input triangle T_{ijk} in M and (w'_l, w'_m, w'_n) is the corresponding point on a triangle T'_{lmn} in mesh M'.

4.5.1 Cusped Hyperbolic Metric on Meshes

The central idea of the theory in [Gu et al. 2018b] and several other papers dealing with related problems is a construction of a hyperbolic metric corresponding to a Euclidean metric $\boldsymbol{\ell}$ which is invariant to conformal scale factors $\boldsymbol{u}$; in this context the lengths $\boldsymbol{\ell}$ are referred to as *Penner coordinates* of the hyperbolic metric.

Conformal deformations of $\boldsymbol{\ell}$ do not change this hyperbolic metric, and flips define just different triangulations of a fixed surface. The update of Penner coordinates for an edge flip using the Ptolemy formula 4.6 happens to produce a mesh that is isometric in the hyperbolic metric to the mesh before the flip. Next, we discuss the hyperbolic metric definition and isometric retriangulation in this metric in more detail.

Beltrami-Klein Model. We use the *Beltrami-Klein* hyperbolic plane model. The model represents the hyperbolic plane H^2 as the interior of a unit disk, with points of the boundary of the disk being points at infinity in the hyperbolic metric. These points (which are not a part of the hyperbolic plane, but play an important role in the model) are called *ideal points*. The model has the following properties.

- Lines are segments connecting points on the boundary.
- Given two distinct points p and q in the disk, the unique Euclidean straight line connecting them intersects the disk's boundary at two ideal points, a and b; label them so that the points are, in order, a, p, q, b along the line. The hyperbolic distance between p and q then is:
$$d_H(p,q) = \frac{1}{2}\log\frac{|aq|\ |pb|}{|ap|\ |qb|}$$
- *Isometries of the hyperbolic plane correspond to projective transformations preserving the unit disk.*
- An isometry is defined uniquely by specifying images of three points on the boundary of the disk (ideal points). There is an isometry mapping any three ideal points to any other three ideal points. We denote such projective maps $P[T \to T']$ where T and T' are triples of points on the unit disk (4.6 center).
- While angles are not preserved, if a line is a diameter, perpendicular lines are also perpendicular to it in the model.

Defining the Hyperbolic Metric. For a mesh M with vertices excluded, the hyperbolic metric is defined by mapping each mesh triangle, with edge lengths given by $\boldsymbol{\ell}$, to a similar Euclidean triangle inscribed in a unit disk, and using the Beltrami-Klein model to define the hyperbolic distances inside the triangle. Under this hyperbolic metric the triangles are ideal, with vertices at infinity (referred to as *cusped*, for reasons more obvious in the Poincaré model, see 4.6 left). Furthermore they are all congruent, because there is a hyperbolic isometry, a projective circumcircle-preserving map, mapping one triangle to the other.

Note however, that unlike the case of finite triangles, the identification of sides of ideal triangles that are adjacent in M is not unique: because the sides are infinitely long, one can slide

them along each other isometrically. The natural gluing defined by identifying points that correspond in the disk model picks one such isometric identification. One can show that if Penner coordinates $\boldsymbol{\ell}$ and $\tilde{\boldsymbol{\ell}}$ are related by a set of conformal scale factors $\boldsymbol{u}$, the resulting gluing between adjacent ideal triangles is the same, i.e., they define the same metric.

This allows a convenient definition of *two-triangle isometric charts* (4.6, right) for this metric, which provide most of what we need for defining our maps f across edge flips.

Two-Triangle Charts. Consider two adjacent triangles T_{ijk} and T_{jil} sharing edge e_{ij}, and five Penner coordinates ℓ_{ij}, ℓ_{jk}, ℓ_{ki}, ℓ_{il}, ℓ_{lj}. For a single triangle, Penner coordinates can be changed arbitrarily using conformal deformations. Note however, that there are only four conformal scale factors u_i, u_j, u_k, u_l involved when mapping two adjacent triangles, so the five lengths cannot be chosen completely arbitrarily. The invariant that is preserved under these remappings is the *cross-ratio* $c_{ij} = (\ell_{jk}\ell_{il})/(\ell_{jl}\ell_{ki})$. Cross-ratio assignments to edges (*shear coordinates*) actually are in one-to-one correspondence with choices of cusped hyperbolic metrics on a fixed mesh.

We are thus free to choose the conformal scale factors u_i, u_j, u_k, u_l so that the following conditions are satisfied: (1) edge e_{ij} is mapped to the diameter $(-p, p)$, with $p = (1, 0)$, on the horizontal coordinate axis; (2) vertices k and l are mapped to antipodal points $p^r = (r, \sqrt{1-r^2})$ and $-p^r$ on the circle. It is easy to check that the four scale factors are uniquely defined by these conditions, with r equal to $(1 - c_{ij})/(1 + c_{ij})$. Notice that $c_{ij} > 0$, thus $r \in (-1, 1)$, regardless of any triangle inequality condition. We denote these two chart triangles T_1^r and T_2^r.

Thus, an isometric atlas can be constructed for the whole mesh, by mapping each triangle pair to a chart as described above. This gives us the necessary tools to define the map f.

Mapping Across a Flip. Let M_k be a mesh obtained after applying a sequence of k flips to M, and M_{k+1} a mesh obtained by flipping a single further edge e_{ij} shared by triangles $T_1 = T_{ijk}$ and $T_2 = T_{jil}$ as above. Each mesh has length assignments $\boldsymbol{\ell}_k$, but as these are guaranteed to satisfy triangle inequalities only at certain steps k where the Delaunay condition is satisfied, these are

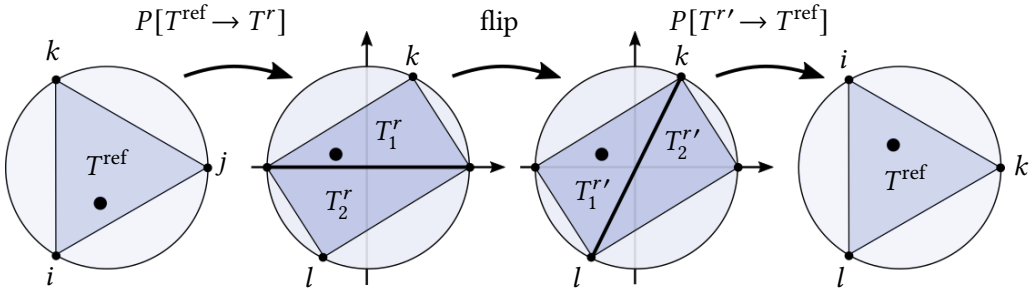


Figure 4.7: Mapping a point through a single flip via a two-triangle chart.

best viewed as Penner coordinates for a hyperbolic metric.

As barycentric coordinates are not invariant with respect to projective maps, we need to choose a reference triangle for barycentric representation (w_i, w_j, w_k). We use an equilateral reference triangle T^{ref}, with vertices q_0, q_1, q_2, with $q_s = (\cos 2s\pi/3, \sin 2s\pi/3)$ for any triangle T_1 of M_k, see 4.7 left.

In the two-triangle chart, T_1 and T_2 are mapped to T_1^r and T_2^r. After the flip in the chart, the new chart triangles, corresponding to triangles $T_1' = T_{jkl}$ and $T_2' = T_{ilk}$ are $T_1^{r\prime} = (p, p^r, -p^r)$ and $T_2^{r\prime} = (-p, -p^r, p^r)$, see 4.7 center. If the image of the point (w_i, w_j, w_k) in the chart belongs to triangle T_1' then the map $(w_i, w_j, w_k) \to (w_i', w_j', w_k')$ is obtained as the composition of circumcircle-preserving projective maps:

$$\begin{aligned}(w_j', w_k', w_l') &= f(w_i, w_j, w_k) = \\ &\left(P[T_1^{r\prime} \to T^{\text{ref}}] \circ B \circ P[T^{\text{ref}} \to T_1^r]\right)(w_i, w_j, w_k)\end{aligned} \tag{4.9}$$

where B is the matrix converting barycentric coordinates on T_1^r to barycentric coordinates on $T_2^{r\prime}$. The expression is similar in the case when the image of the point in the chart lands in $T_2^{r\prime}$. The circumcircle-preserving projective maps P can be computed in barycentric coordinates using the following formula:

$$P(w_i, w_j, w_k) = (w_i S_i, w_j S_j, w_k S_k)/(w_i S_i + w_j S_j + w_k S_k)$$

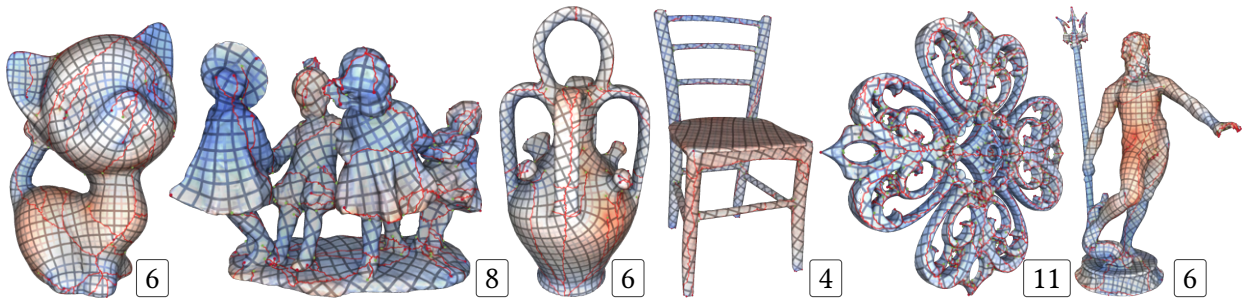


Figure 4.8: Visualization of conformal maps, implied by conformal cone metrics, on some of the closed models with angle prescriptions from the dataset of [Myles et al. 2014]. The numbers indicate the scale range (difference of maximal and minimal conformal (natural) logarithmic scale factor $\boldsymbol{u}$) for each model. Cones are marked by red and green dots; texture jumps due to cones are marked red. The textured map and scale visualization follow the description from 4.6.

with $S_i = \frac{\ell_{ij}\ell_{ki}\tilde{\ell}_{jk}}{\tilde{\ell}_{ij}\tilde{\ell}_{ki}\ell_{jk}}$, where $\boldsymbol{\ell}$ are lengths of the source, and $\tilde{\boldsymbol{\ell}}$ are lengths of the target triangle.

4.6 Evaluation

We have implemented 3 (with support for boundaries following 4.4) in C++. Our goal is to assess how well this theoretically sound method performs practically. While by default we use standard double precision floating point numbers, the optional use of extended precision arithmetics in our implementation allows us to assess to what extent potential convergence issues are related to finite precision or other problems, as detailed in 4.6.3, 4.6.4. We find that, as conformal maps can easily involve a very large range of scales across a mesh, for certain challenging settings the use of extended precision arithmetics can be essential to yield results of adequate quality.

In cases where a (mostly) flat metric is computed, the result can be visualized by turning the metric into a map (using a layout of the flat mesh in the plane [Springborn et al. 2008]) and mapping a texture (e.g. a grid or checkerboard) to the surface using this map. For a clear visualization in cases with high scale distortion, we use a procedurally generated hierarchical grid texture, as illustrated here. Its density is chosen adaptively based on the pointwise magnitude of the scale distortion on the surface mesh, halving the spacing between texture lines when the scale

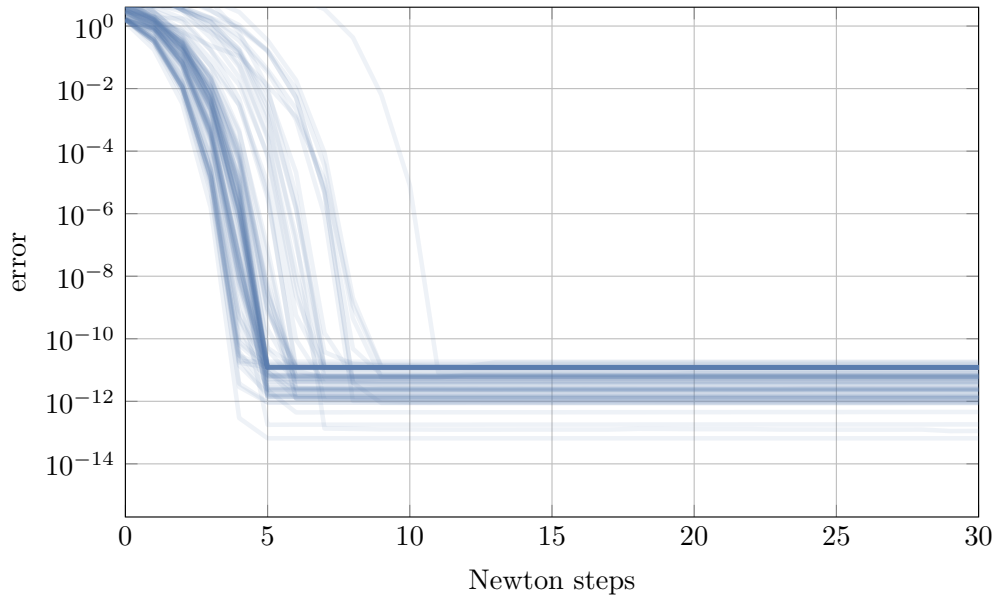


Figure 4.9: Decay of maximum angle error $\|\hat{\Theta} - \Theta\|_\infty$ over the iterations of the Newton algorithm. Each graph represents one of the closed-surface instances from the dataset of [Myles et al. 2014].

factor of the conformal map is halved.

4.6.1 Validation

Closed Surfaces. A dataset of mesh models together with angle prescriptions $\hat{\Theta} > 0$ (based on cones of cross fields) has been released with [Myles et al. 2014]. We applied our implementation to the closed models from this dataset. The angle error decay in the course of the algorithm on these cases is visualized in 4.9. Some of the models with the resulting conformal map are visualized in 4.8. We observe that the models reach angle accuracy of 10^{-10} in less than 15 Newton iterations. The final achievable accuracy varies and is correlated with the range of scale factors in the final mesh (cf. 4.20), as a large variation of scale factors leads to a moderate loss of precision in the gradient computation.

As further test instances, we use 1000 different random target angle prescriptions $\hat{\Theta}$ (with $\hat{\Theta}_i \in (\pi, 3\pi)$ for all vertices v_i) on a sphere mesh (1K vertices). The error decay is visualized in 4.10. Note that the overall behavior is very similar, whether the prescribed angles are random or

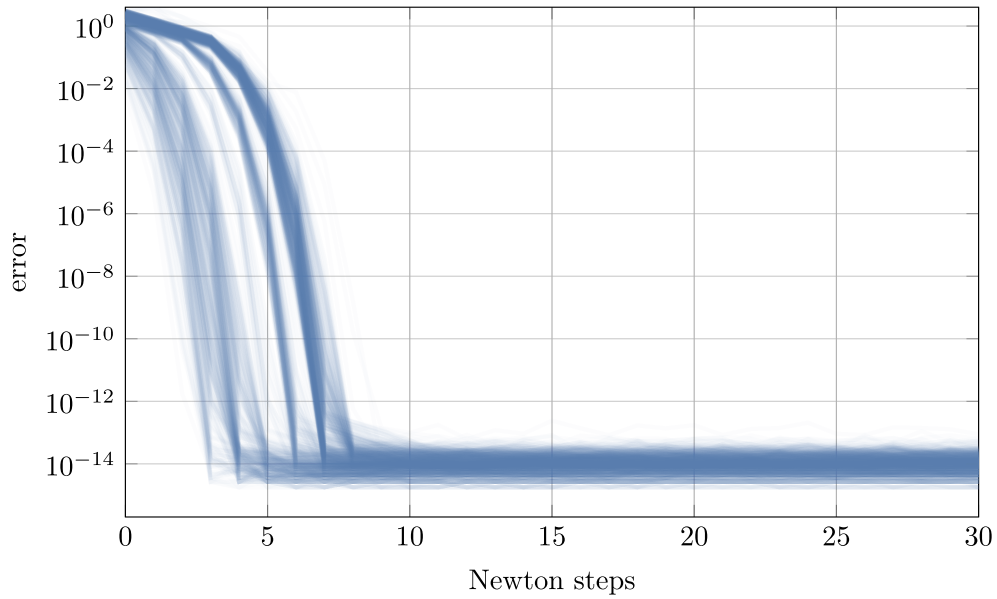


Figure 4.10: Like 4.9, but each graph represents one of 1000 random test instances (again without boundary)

geometrically meaningful (as in 4.9).

We consider the extreme scenario of concentrating the target metric's entire curvature in one point (i.e., prescribing a single cone of angle $2\pi(2g-1)$ in an otherwise flat metric). Errors for surfaces of increasing genus g (procedurally generated g-tori) are shown in 4.11. A blow-up of the configuration around the single prescribed cone vertex on a genus 6 example is shown in 4.13.

Surfaces with Boundary. The above mentioned dataset from [Myles et al. 2014] also contains meshes with one or more boundary loops, together with angle prescriptions $\hat{\Theta} > 0$ for interior and boundary vertices. The error decay on these cases is shown in 4.14. Some of the models are visualized in 4.12. The behavior is overall similar to the closed surface case.

As a synthetic test, we generate 1000 different random target angle prescriptions $\hat{\Theta}$ for a surface with boundary (a disk with 5K vertices). In the interior we prescribe a flat metric, at the boundary we prescribe a geodesic curvature, maximally in the range $\pm\pi$, i.e., $\hat{\Theta}_i \in (0, 2\pi)$ for all boundary vertices v_i. 4.15 shows the number of the different types of symmetric flips that are performed in the course of the algorithm on these cases. As expected, the number of flips is larger

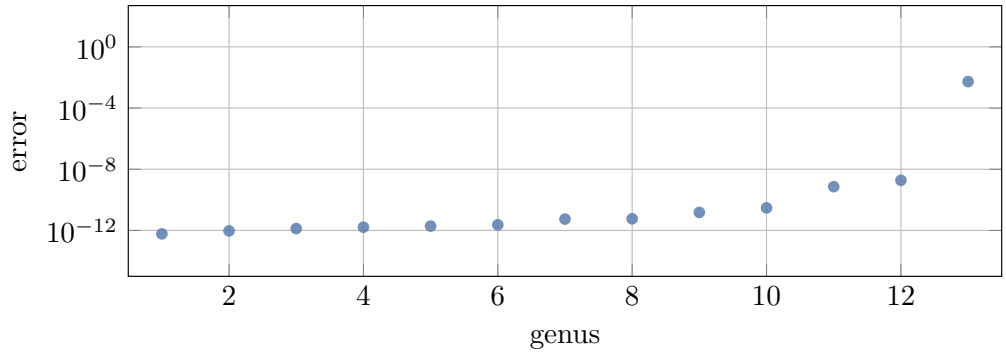


Figure 4.11: Final residual angle error for the extreme case of concentrating all curvature in a single cone on an g-torus surface (genus g). For the genus 12 case, where the residual error is still benign, the conformal scale factor spans 232 orders of magnitude. For the problematic genus 13 case it surpasses 262. By increasing numerical precision (4.6.3), this can be remedied; for instance, with 200-bit precision, the $g = 150$ case converges to below 10^{-29}, with 400-bit precision, the $g = 400$ case to below 10^{-65} (with the scale factors spanning 611 orders of magnitude). (To reduce numerical issues in this extreme experiment, the initial step size λ was halved until the range of the coefficients of $\lambda \boldsymbol{d}$ was less than 10.)

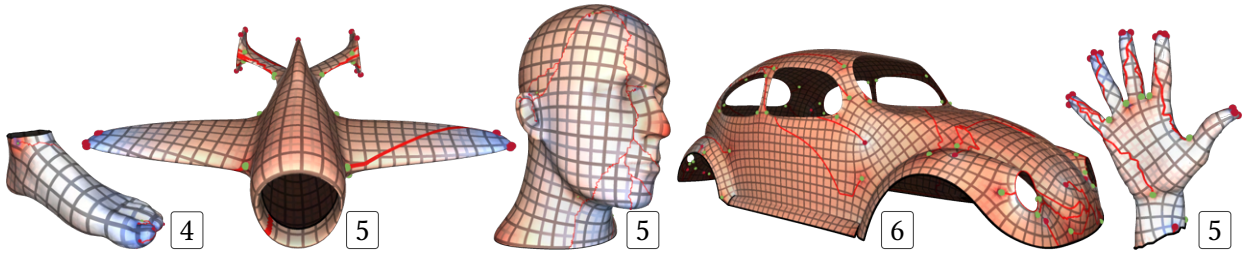


Figure 4.12: Visualization of conformal maps, analogous to 4.8, on some of the models *with boundary* from the dataset of [Myles et al. 2014]. The boundary geodesic curvature is prescribed to be zero, therefore the angle between texture grid lines and the boundary is constant per boundary loop.

for cases with a prescribed curvature spanning a larger range.

Another relevant scenario is that of prescribed geodesic curvature along a cut graph. We take the closed models of the dataset from [Myles et al. 2014] and mimic the setting employed by [Campen et al. 2019]: we compute a cut graph on each of these surfaces, and prescribe $\hat{\Theta}_i = \pi$ along this cut graph's segments' from both sides (effectively asking them to be straight under the conformal metric). The cut graph is composed of g short handle loops computed as in [Diaz-Gutierrez et al. 2009], connected by additional shortest paths. The resulting cut forms a graph on the surface with nodes of valence 3; at each node, an angle of π is prescribed for the largest sector, and angles $\pi/2$ for the remaining two. Some of the models are visualized in 4.16, with the

cut graph marked in black. Depending on the shape of the cut graph, this scenario turns out to be the most challenging numerically: As can be seen in 4.17 left, in a few cases the final maximum error is over 10^{-10}, i.e., higher than in the previously discussed scenarios. This is related to the scale distortion of the implied conformal metric spanning a range of up to 73 orders of magnitude in these cases. With higher-precision arithmetic, these residuals can be reduced, as discussed in 4.6.3.

4.6.2 Comparison

We demonstrate the advantages of the Delaunay flip approach over the degeneration flip approach (4.2.2) in terms of efficiency as well as numerical robustness. To this end, we apply an implementation of the described method and an implementation of the algorithm described by [Campen and Zorin 2017b] (both using standard double precision floating point numbers) to the same set of inputs.

Efficiency. The main differences between the two methods lie in the number of linear system solves (to compute the descent direction $\boldsymbol{d}$) and the number of intrinsic flips. In the proposed method, the number of flips is often significantly higher (see the discussion in 4.2.2), while the number of system solves is lower. As a flip is a cheap local operation, while a system solve is an expensive global operation, a run time benefit can be conjectured.

The scatter plot in 4.18 shows that this is the case on average. As test instances we use 1000 different random target angle prescriptions $\hat{\boldsymbol{\Theta}}$ (with $\hat{\Theta}_i \in (\pi, 3\pi)$ for all vertices v_i) on a sphere mesh (10K vertices). Only for relatively simple cases, where the target curvature can be matched without degeneration flips, the number of system solves may be similar. On average, run time is 73× lower with the Delaunay-based method on these examples.

Robustness. Differences in robustness can best be observed by considering extreme cases. In 4.19 we show the residual error of the two methods when prescribing one very small or very large target angle (while distributing the remaining curvature). For small angles it becomes apparent that the degeneration flip algorithm is numerically more fragile.

4.6.3 Accuracy

While the method is theoretically guaranteed to yield the desired result, in practice numerical inaccuracies limit how closely the target curvature will be matched. As the method involves exponential and trigonometric functions (4.1, 4.2), it cannot be implemented in a numerically exact manner using adaptive precision rational or integer number types. Using extended precision floating point number types (such as MPFR), the method's accuracy can, however, be increased arbitrarily. We evaluate the effect of this choice on result accuracy in 4.20. As test instances we use 1000 different random target angle prescriptions $\hat{\Theta}$ (with $\hat{\Theta}_i \in (\pi, 3\pi)$ for all vertices v_i) on a sphere mesh (1K vertices).

As can be observed, the remaining error decreases consistently as the number of bits used for the floating point computations is increased. Due to dependence on many factors (input mesh and edge lengths, target angles, choice of linear system solver for the Newton direction) a simple bound on the error cannot be given, but 4.20 gives an empirical idea of the behavior. Note that some correlation can be observed to the conformal scale distortion (the range $[e^{\min \boldsymbol{u}}, e^{\max \boldsymbol{u}}]$) that is required to match the target curvature.

In 4.17 right the effect of increased precision on test cases from 4.6.1 can be observed. In particular, for the models that have maximum error over 10^{-10} when using standard double precision arithmetic, the error is reduced to below 10^{-16} when using a 100 bits mantissa instead.

4.6.4 Failure Modes

We can distinguish two types of potential issues (detailed below): high residual error or a high number of optimization steps. The former can be caused by limited arithmetic precision (and can therefore be remedied by using extended precision, as demonstrated above). The latter can be caused by an unfavorable energy landscape, and is therefore more fundamental, regardless of numerics.

Precision-related failures. Depending on the target curvature, a high amount of metric distortion may be required, with negative numerical effects on result accuracy. It can be observed that this is correlated with local concentrations of positive or negative target (Gaussian or geodesic) curvature. Figure 4.21 left shows an experiment in which an increasingly large cluster of vertices have a target angle below 2π and the rest above 2π. When using standard double precision, a large fraction of these synthetic test cases essentially fails to reach a reasonably accurate state. Performing these computations with higher precision (Figure 4.21 right) resolves these problems. Analogously, we notice that cut graphs with more complex shape than the ones used in 4.17 (e.g., the more constrained "hole-chain" in [Campen et al. 2019]) cause a similar behavior.

Near-degeneracy failures. This second issue is more fundamental. While the method may, in principle, converge eventually, the step size can decrease to the point that the number of iterations needed becomes impractical. When using the above mentioned hole-chain choice of cuts on the dataset of [Myles et al. 2014], we can identify four high-genus models with complex singularities which fail to converge in a reasonable number of steps even with high-precision arithmetic. The underlying reason is illustrated in Figure 4.22, showing the plot of the projected gradient $\boldsymbol{d}^\top \boldsymbol{g}(\boldsymbol{u} + \lambda \boldsymbol{d})$ for a line search direction. One can see that while theoretically the gradient is C^1, it may experience very significant jumps, when a large number of triangle flips happen nearly simultaneously as λ changes (in this particular case 58). We observe that this occurs in particular

in the presence of highly distorted near-degenerate triangles.

4.7 Proofs and additional Lemmas

Proof of Proposition 2

Proof. If x is not fixed, by the well-definedness of R on mesh elements, for each $h \in x$ we have $R(h) \notin x$. Therefore for a non-fixed individual face or edge x all its halfedges can be assigned to H^1 (or to H^2) without contradicting the conditions. It needs to be shown that this can be done for all such elements consistently.

Let H^e the set of halfedges whose edges are not fixed and H^f the set of halfedges whose faces are not fixed. Let Q the relation that is the union of $O|_{H^e}$ and $N|_{Hf}$ on $H \setminus H^s$. Consider the connected components H_i of Q (intuitively: the mesh's connected components separated by fixed edges and fixed faces). Due to the properties of R (preserving/inverting O and N) it is well-defined on these connected components via $R(H_i) = H_j \Leftrightarrow R(h) \in H_j$ for any $h \in H_i$. Using arguments analogous to [Panozzo et al. 2012, Prop. 2] one verifies that the set of fixed elements necessarily forms a cycle; therefore there are at least two such connected components.

As R on $H \setminus H^s$ has orbits of length 2 only, it allows a bipartition of the connected components, i.e., they can be assigned to two sets H^1 and H^2 in accordance with the above conditions. □

Label Compatibility

Proposition 2.

(a) $e \in E^{\perp} \Rightarrow f_a, f_b \in F^s$.

(b) $e \in E^{\parallel} \Rightarrow f_a \in F^1, f_b \in F^2$ *or* $f_a = f_b \in F^s$.

(c) $e \in E^1 \Rightarrow f \notin F^2,\ e \in E^2 \Rightarrow f \notin F^1$.

(d) $e \in E^i, f_a, f_b \in F^s \Rightarrow R(e) \in f_a, f_b$.

Proof. Part (a) follows immediately from the definition of F^i, as faces from F^i cannot have edges from $E^\perp$.

Suppose a face f_a is incident at an edge e from $E^\parallel$. For these edges $R(e) = e$. Suppose $f_a \in F^1$, then $R(f_a)$ is incident to $R(e) = e$, therefore $f_b = R(f_a)$. As $R(f_a) \in F^2$ by definition of F^2, this proves the first part of (b). Suppose $f_a \in F^s$, and let h a halfedge $h \in e$, $h \in f_a$. Then $R(h) \in f_a$ by the definition of F^s; but, by definition of $E^\parallel$, $R(h) \in e$, so $f_a = f_b$, i.e., a face is adjacent to itself along e.

Part (c) directly follows from the definitions of E^i and F^i.

In part (d), suppose f_a and f_b are incident at $e \in E^1$, $f_a, f_b \in F^s$, and $e = (h_a, h_b)$. Then $R(h_a) \in R(f_a) = f_a$, $R(h_b) \in f_b$, and $O(R(h_a)) = R(h_b)$ by the properties of R, i.e., $(R(h_a), R(h_b))$ is an edge. By definition of E^i, it has to be in E^2, i.e., faces f_a and f_b share a second edge, and this edge is from E^2. □

Irrelevance of Flip Types $(s, \|, s)$ and $(s, 1, s)$

Proposition 3. *Types* $(t, \|, t)$, $(q, \|, q)$, $(t, 1, t)$, $(t, 1, q)$, *and* $(q, 1, q)$ *are associated with edges that are Delaunay regardless of metric.*

Proof Consider $(t, \|, t)$. By 2(b), it corresponds to a configuration with a single face: $(f^t, e^\parallel, f^t)$. As the triangle f^t is isosceles, and both side edges of the triangle coincide with $e^\parallel$, angles opposite $e^\parallel$ are $\pi/2 - \alpha/2$ if the apex angle is α, i.e., their sum is guaranteed to be less than π and the edge is Delaunay. For $(q, \|, q)$, to evaluate the Delaunay criterion, we split f^q into triangles. As f^q is inscribed the choice of diagonal does not affect the angles; we can choose the diagonal that connects a vertex of $e^\parallel$ with a vertex with trapezoid angles $\leq \pi/2$, from which we can see that both angles opposite $e^\parallel$ are less than $\pi/2$. For cases $(t, 1, t)$, $(t, 1, q)$, and $(q, 1, q)$ the same logic applies to each face incident at the shared edge e^1. □

4.8 Double Cover: Formal Definition

Given a mesh $N = (H^0, \mathcal{N}^0, O^0)$, with boundary and interior halfedges $H^{bnd} \cup H^{int} = H^0$, we discard H^{bnd} and set $H = H^1 \cup H^2$ where $H^1 = H^{int}$ and $H^2 = \bar{H}^{int}$, where $\bar{\cdot}$ denotes a copy. The reflection map R is defined via $R(h) := h'$ if $h' \in H^2$ is the copy of $h \in H^1$. O^0 is adopted on both copies to define O, except that $O(h) := R(h)$ if $O^0(h) \in H^{bnd}$; this latter adjustment constitutes the *gluing* of the two copies along their boundaries. Finally

$$\mathcal{N}(h) := \begin{cases} \mathcal{N}^0(h) & \text{if } h \in H^1, \\ R((\mathcal{N}^0)^{-1}(R(h))) & \text{if } h \in H^2. \end{cases}$$

This forms the symmetric double cover mesh $M = (H, \mathcal{N}, O, R)$ with triangle faces and map R. Note that R is a reflection map: it satisfies the conditions of theorem 4.6 (where condition (3) is void as M has no boundary). It is easy to see that this construction implies $E^{\perp} = \varnothing$ and $F^s = \varnothing$, i.e., no element crosses the symmetry line (the former boundary). $E^{\parallel}$ contains the edges lying *on* the symmetry line, i.e., those for whose halfedges the O relation was adjusted to glue the two copies.

4.9 Conclusions and Future Work

We presented a practical realization of the method for computing discrete conformal maps based on the ideas of [Gu et al. 2018b; Springborn 2019], elaborating how it can be applied safely to meshes with boundary, the most practically relevant scenario for conformal mapping. Our improvements include a straightforward to implement algorithm for maintaining symmetric Delaunay triangulations and several improvements increasing the robustness of Newton's optimization method in the context of our application. We explored its behavior on a standard dataset, and for

a number of challenging synthetic examples, demonstrating its robustness for a broad range of cases involving high distortion. We also observe that common failure cases can be addressed by using extended precision arithmetic, albeit at a significant cost in run time.

However, in our extensive tests we did identify a small number of cases for which the method does not produce a conformal map in reasonable time, which indicates potential for further algorithmic improvements. It would also be desirable to find ways to minimize the use of extended precision arithmetic to the minimum necessary in a *filtered* approach, so as to increase accuracy while maintaining performance. Finally, extension of the method from Euclidean to spherical and hyperbolic discrete metrics would be not only of theoretical interest [Schmidt et al. 2020].

Algorithm 3: FindConformalMetric

Input : triangle mesh $M = (V, E, F)$, closed, manifold, edge lengths $\boldsymbol{\ell} > 0$ satisfying triangle inequality, target angles $\hat{\boldsymbol{\Theta}} > 0$ respecting Gauss-Bonnet

Output: triangle mesh $M' = (V, E', F')$, edge lengths $\tilde{\boldsymbol{\ell}} > 0$ satisfying triangle inequality, such that $\|\boldsymbol{\Theta}_{(M', \tilde{\boldsymbol{\ell}})} - \hat{\boldsymbol{\Theta}}\|_\infty \leq \varepsilon_{\text{tol}}$

Function *FindConformalMetric*$(M, \boldsymbol{\ell}, \hat{\boldsymbol{\Theta}})$**:**
  $\boldsymbol{u} \leftarrow \mathbf{0}$, $(M, \boldsymbol{\ell}) \leftarrow$ MakeDelaunay$(M, \boldsymbol{\ell}, \boldsymbol{u})$
  while not *converged*$(M, \boldsymbol{\ell}, \boldsymbol{u})$ **do**
    $\boldsymbol{g} \leftarrow g(M, \boldsymbol{\ell}, \boldsymbol{u})$; $H \leftarrow H(M, \boldsymbol{\ell}, \boldsymbol{u})$ // gradient and Hessian
    $\boldsymbol{d} \leftarrow solve(H\boldsymbol{d} = -\boldsymbol{g})$ // Newton direction
    $(M, \boldsymbol{\ell}, \boldsymbol{u}) \leftarrow$ LineSearch$(M, \boldsymbol{\ell}, \boldsymbol{u}, \boldsymbol{d})$ // Newton step
  $\tilde{\boldsymbol{\ell}} \leftarrow$ScaleConformally$(M, \boldsymbol{\ell}, \boldsymbol{u})$
  return $(M, \tilde{\boldsymbol{\ell}})$

Function *LineSearch*$(M, \boldsymbol{\ell}, \boldsymbol{u}, \boldsymbol{d})$**:**
  $(M_1, \boldsymbol{\ell}_1) \leftarrow$ MakeDelaunayPtolemy$(M, \boldsymbol{\ell}, \boldsymbol{u} + \boldsymbol{d})$
  $(M_{1/2}, \boldsymbol{\ell}_{1/2}) \leftarrow$ MakeDelaunayPtolemy$(M, \boldsymbol{\ell}, \boldsymbol{u} + \frac{1}{2}\boldsymbol{d})$
  if $\frac{1}{2}\left(\boldsymbol{d}^\intercal g(M_1, \boldsymbol{\ell}_1, \boldsymbol{u}+\boldsymbol{d}) + \boldsymbol{d}^\intercal g(M_{1/2}, \boldsymbol{\ell}_{1/2}, \boldsymbol{u}+\frac{1}{2}\boldsymbol{d})\right) \leq \alpha \boldsymbol{d}^\intercal g(M, \boldsymbol{\ell}, \boldsymbol{u})$ **then**
    return $(M_1, \boldsymbol{\ell}_1, \boldsymbol{u} + \boldsymbol{d})$ // full step
  while *true* **do** // line search
    $(M, \boldsymbol{\ell}) \leftarrow$ MakeDelaunayPtolemy$(M, \boldsymbol{\ell}, \boldsymbol{u} + \boldsymbol{d})$
    if $\boldsymbol{d}^\intercal g(M, \boldsymbol{\ell}, \boldsymbol{u} + \boldsymbol{d}) \leq 0$ **then** // 4.7
      return $(M, \boldsymbol{\ell}, \boldsymbol{u} + \boldsymbol{d})$
    $\boldsymbol{d} \leftarrow \frac{1}{2}\boldsymbol{d}$ // backtracking

Function $g\ (M, \boldsymbol{\ell}, \boldsymbol{u})$**:**
  return $\hat{\boldsymbol{\Theta}} - \Theta(M, \tilde{\boldsymbol{\ell}})$ // 4.2

Function $H\ (M, \boldsymbol{\ell}, \boldsymbol{u})$**:**
  return CotanLaplacian$(M, \tilde{\boldsymbol{\ell}})$

Function $\Theta(M, \boldsymbol{\ell}, \boldsymbol{u})$**:** // angle computation
  for $v_i \in V$ **do** // 4.2
    $\Theta_i \leftarrow \sum_{T_{ijk} \in M'} \arccos\left((\tilde{\ell}_{ij}^2 + \tilde{\ell}_{ki}^2 - \tilde{\ell}_{jk}^2)/(2\tilde{\ell}_{ij}\tilde{\ell}_{ki})\right)$
  return $(\Theta_0, \ldots, \Theta_n)$

Function *MakeDelaunayPtolemy*$(M, \boldsymbol{\ell}, \boldsymbol{u})$**:**
  while *NonDelaunay*$(M, \boldsymbol{\ell}, \boldsymbol{u}, e_{ij})$ *for any edge* e_{ij} **do**
    $(M, \boldsymbol{\ell}) \leftarrow$ PtolemyFlip$(M, \boldsymbol{\ell}, e_{ij})$
  return $(M, \boldsymbol{\ell})$

Function *NonDelaunay*$(M, \boldsymbol{\ell}, \boldsymbol{u}, e_{ij})$**:**
  return $(\tilde{\ell}_{jk}^2 + \tilde{\ell}_{ki}^2 - \tilde{\ell}_{ij}^2)/(\tilde{\ell}_{jk}\tilde{\ell}_{ki}) + (\tilde{\ell}_{jm}^2 + \tilde{\ell}_{mi}^2 - \tilde{\ell}_{ij}^2)/(\tilde{\ell}_{jm}\tilde{\ell}_{mi}) < 0$ // 4.4

Function *PtolemyFlip*$(M, \boldsymbol{\ell}, e_{ij})$**:**
  $M \leftarrow$ Flip(M, e_{ij})
  $\ell_{km} \leftarrow (\ell_{jk}\ell_{im} + \ell_{ki}\ell_{mj})/\ell_{ij}$ // 4.6
  return $(M, \boldsymbol{\ell})$

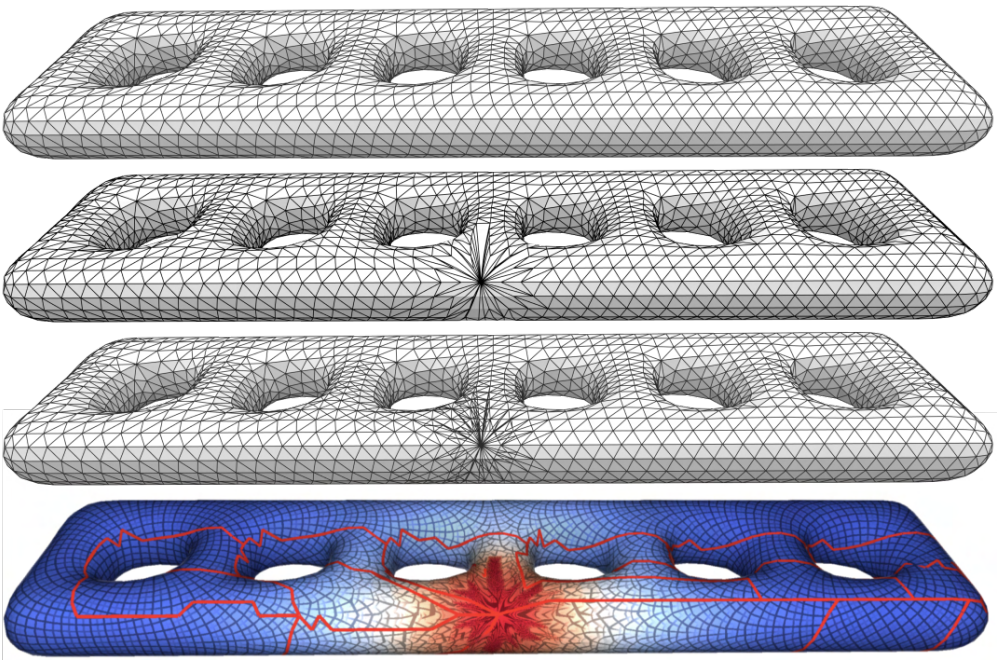


Figure 4.13: Top: Input triangulation. Second row: Resulting intrinsic retriangulation, when concentrating all curvature on a single vertex ($\Theta = 22\pi$); it is Delaunay under the computed conformal metric (with curvature -20π at the central vertex). Third row: overlay triangulation [Fisher et al. 2007], allowing for a simple representation of the implied conformal map, linear or projective per triangle. Bottom: Visualization of implied conformal map using a hierarchical grid texture (spanning 25 levels in this extreme case).

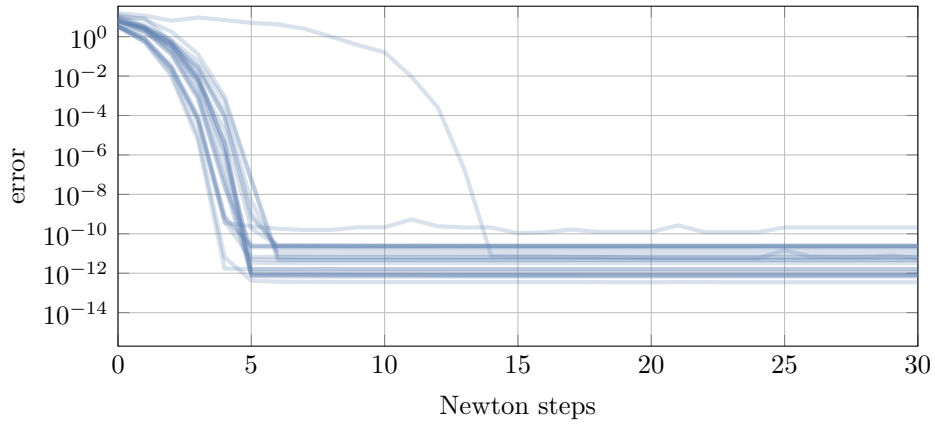


Figure 4.14: Decay of maximum angle error $\|\hat{\Theta} - \Theta\|_\infty$ over the iterations of the Newton algorithm. Each graph represents one of the instances *with boundary* from the dataset of [Myles et al. 2014].

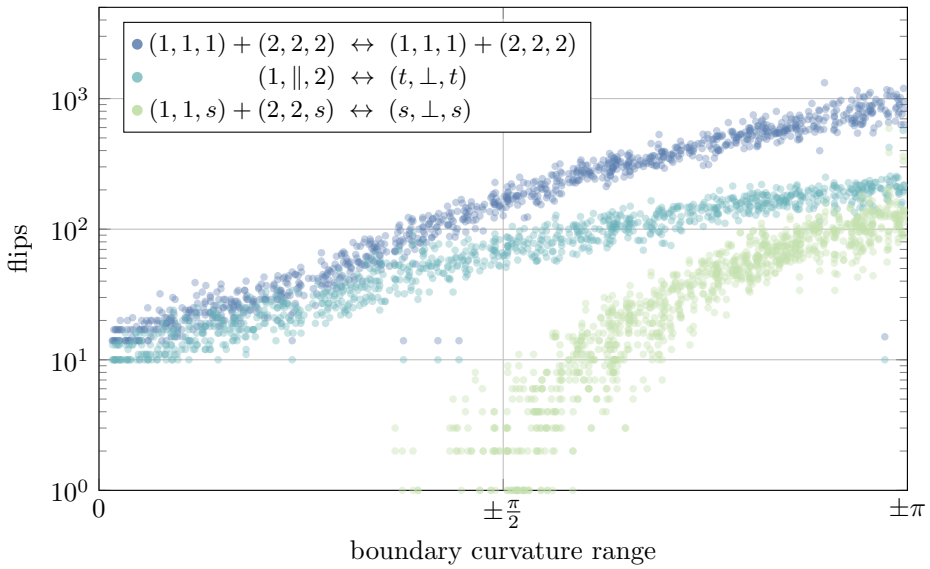


Figure 4.15: Scatter plot showing the numbers of different types of symmetric flips during the algorithm relative to the range of prescribed random boundary curvatures. Each dot represents one type of flips for one of 1000 test instances.

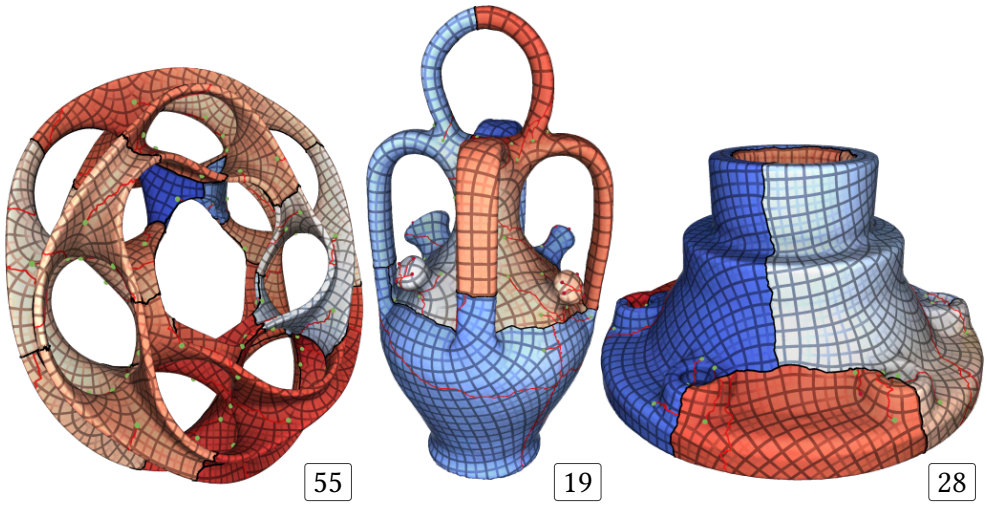


Figure 4.16: Visualization of conformal maps with cones, analogous to 4.8, on models cut to disk topology using a cut graph (black). Due to the prescribed geodesic curvature along the cut boundary, the cut is axis-aligned under the map. Notice that such enforced alignment can easily imply a broad range of scales, which is challenging numerically.

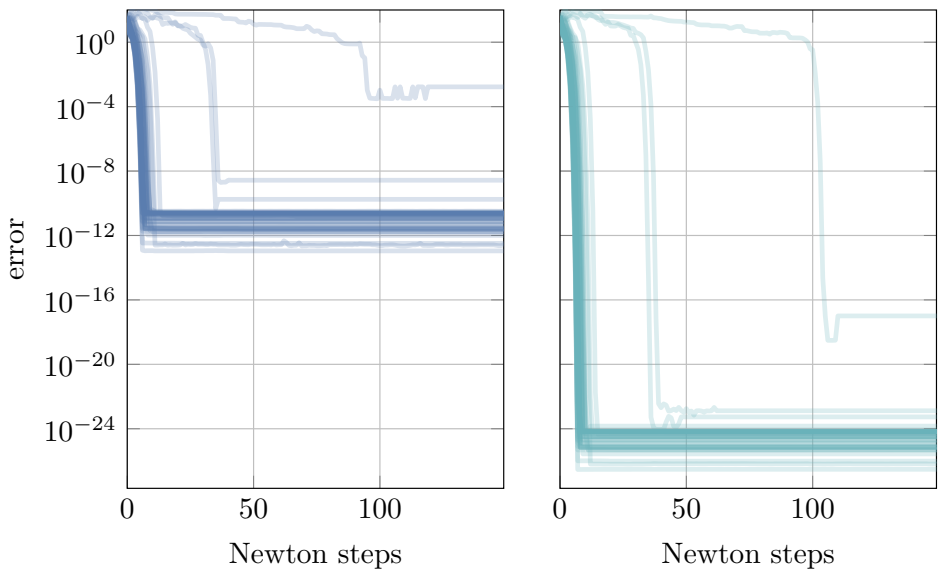


Figure 4.17: Decay of maximum angle error $\|\hat{\Theta} - \Theta\|_\infty$ over the iterations of the Newton algorithm. Each graph represents one of the closed instances from the dataset of [Myles et al. 2014], with *prescribed curvature along a cut graph.* Left: `double` precision. Right: extended precision (100 bits mantissa).

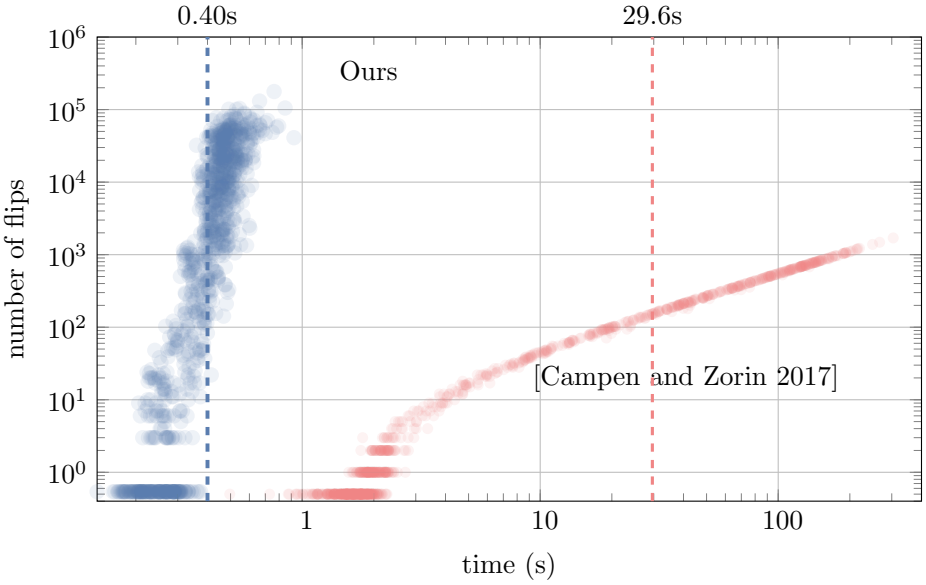


Figure 4.18: Scatter plot showing the number of flips and the run time (to reach $\varepsilon_{\text{tol}} = 10^{-10}$), for the described Delaunay-flip method (blue) and the degeneration flip method (red). Each dot represents one of 1000 test instances. Dashed lines mark the average run time, 0.4s and 29.6s, respectively.

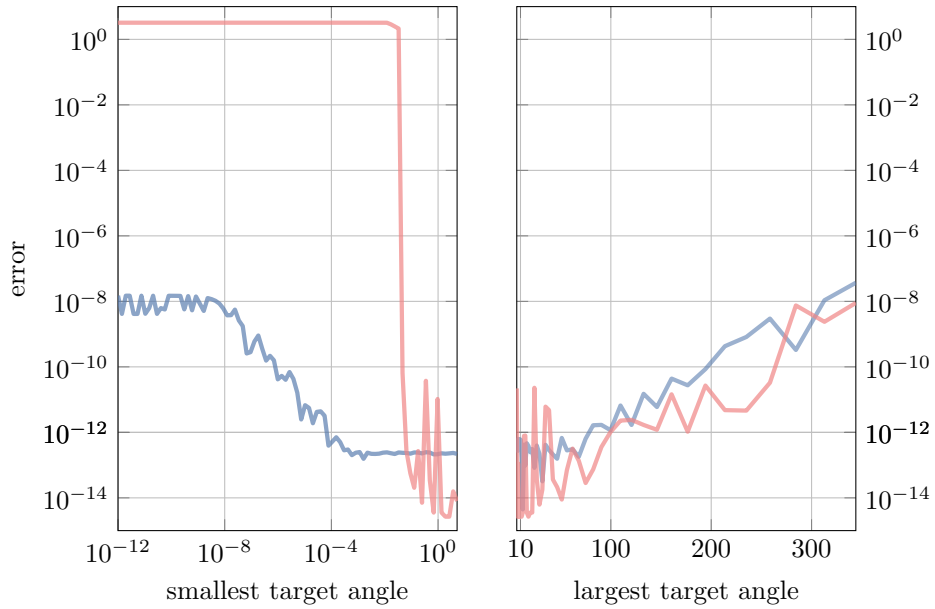
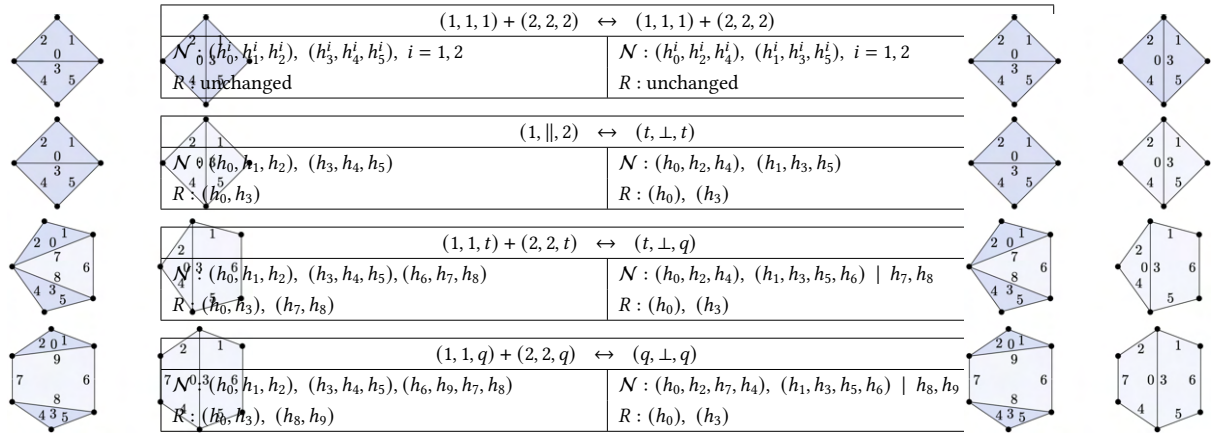


Figure 4.19: Final residual angle error $\|\hat{\Theta} - \Theta\|_\infty$ for extreme cases (one very small or very large target angle, on a sphere with 1K vertices), comparing the Delaunay-based algorithm (blue) and the degeneration flip algorithm. [Campen and Zorin 2017b] (red).

Table 4.1: Combinatorial updates required to perform symmetric flips of all relevant consistent types. The change to $\mathcal{N}$ is given by listing the orbits (halfedge cycles forming faces) of $\mathcal{N}$ created by the flip. The employed indexing is depicted in the figures left and right. Similarly, we define changes to R viewing it as a permutation with orbits of length 1 or 2, and listing the sets of orbits being replaced. Finally, rather than deleting and adding new halfedges on demand, for implementational efficiency we can associate a superfluous pair of halfedges, eliminated by a quad-creating flip, with the quad (listed behind the bar).



$(1,1,1)+(2,2,2)$	$\leftrightarrow$	$(1,1,1)+(2,2,2)$
$\mathcal{N}: (h_0^i, h_1^i, h_2^i),\ (h_3^i, h_4^i, h_5^i),\ i=1,2$ R : unchanged		$\mathcal{N}: (h_0^i, h_2^i, h_4^i),\ (h_1^i, h_3^i, h_5^i),\ i=1,2$ R : unchanged
$(1,\|\|,2)$	$\leftrightarrow$	$(t,\perp,t)$
$\mathcal{N}: (h_0, h_1, h_2),\ (h_3, h_4, h_5)$ $R: (h_0, h_3)$		$\mathcal{N}: (h_0, h_2, h_4),\ (h_1, h_3, h_5)$ $R: (h_0),\ (h_3)$
$(1,1,t)+(2,2,t)$	$\leftrightarrow$	$(t,\perp,q)$
$\mathcal{N}: (h_0, h_1, h_2),\ (h_3, h_4, h_5), (h_6, h_7, h_8)$ $R: (h_0, h_3),\ (h_7, h_8)$		$\mathcal{N}: (h_0, h_2, h_4),\ (h_1, h_3, h_5, h_6)\ \mid\ h_7, h_8$ $R: (h_0),\ (h_3)$
$(1,1,q)+(2,2,q)$	$\leftrightarrow$	$(q,\perp,q)$
$\mathcal{N}: (h_0, h_1, h_2),\ (h_3, h_4, h_5), (h_6, h_9, h_7, h_8)$ $R: (h_0, h_3),\ (h_8, h_9)$		$\mathcal{N}: (h_0, h_2, h_7, h_4),\ (h_1, h_3, h_5, h_6)\ \mid\ h_8, h_9$ $R: (h_0),\ (h_3)$

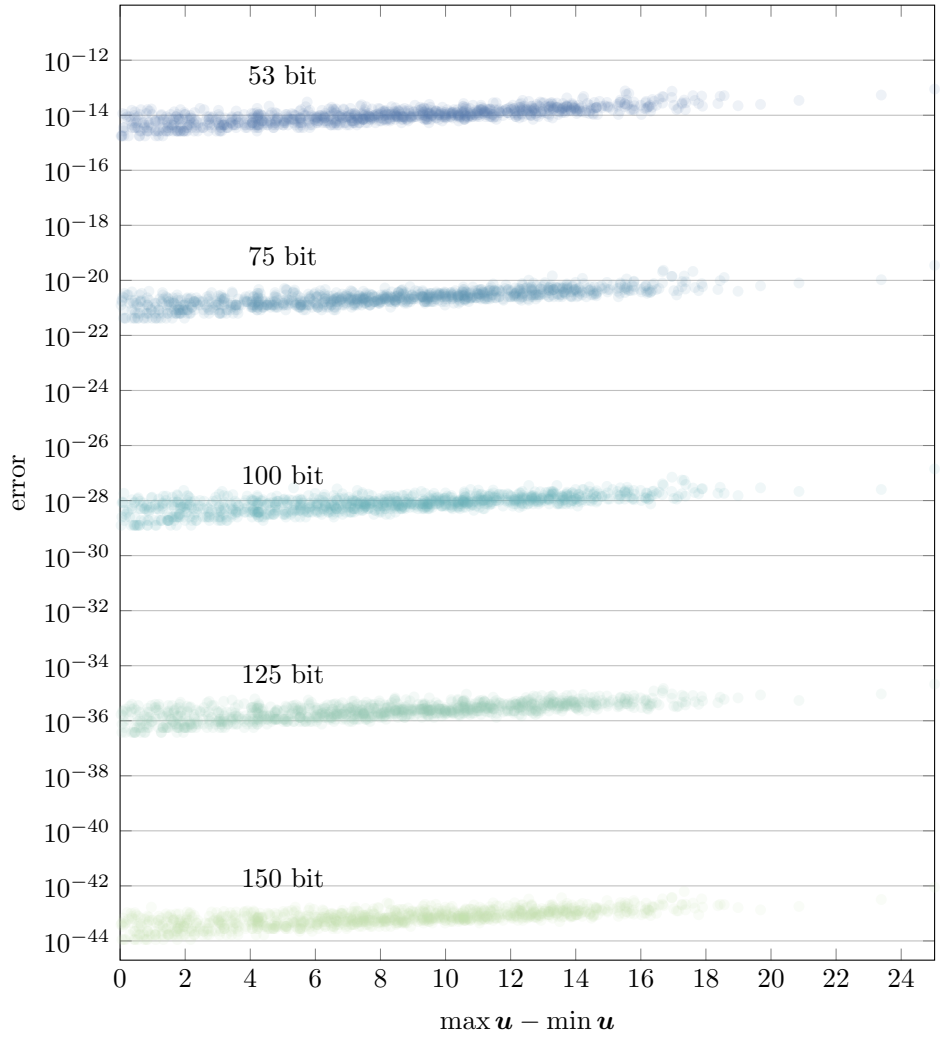


Figure 4.20: Scatter plot showing residual angle error $\|\hat{\boldsymbol{\Theta}} - \boldsymbol{\Theta}\|_\infty$ (after at most 50 Newton steps) relative to the range of logarithmic conformal scale factors $\boldsymbol{u}$. Each dot represents one test instance, run using floating point numbers with a mantissa of 53 bits (`double`), 75 bits, 100 bits, 125 bits, 150 bits (`MPFR`).

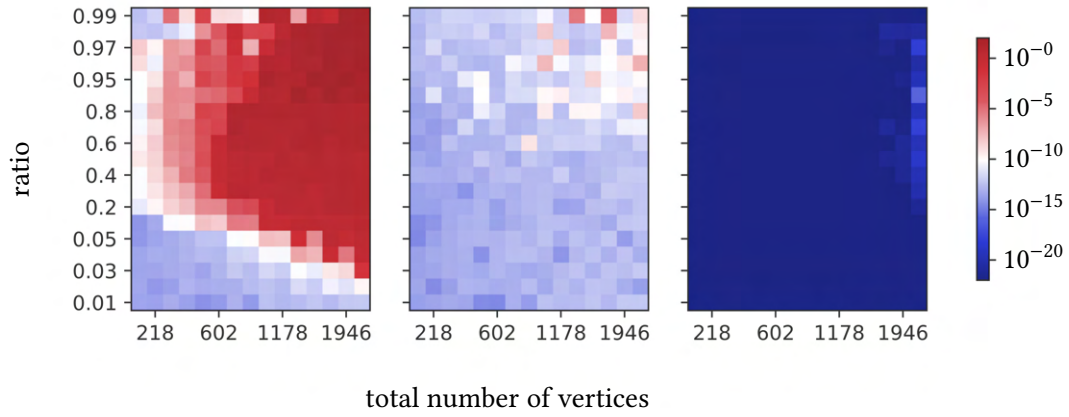


Figure 4.21: Heatmap showing the final error $\|\hat{\Theta} - \Theta\|_\infty$ for spheres of varying resolution (x-axis) with some ratio (y-axis) of the vertices set to target angle 3 and the rest to a constant target angle $< 2\pi$ such that the Gauss-Bonnet theorem is satisfied. Left: double precision results when the two angle values are distributed in two clusters. Center: double precision results when the two angle values are distributed randomly over the sphere. Right: extended precision (150 bits mantissa) results with the same distribution as left. (For this experiment, the threshold for the gradient norm decrease was set to 0 and, to reduce the run time in this particular case, λ was chosen adaptively, initially halved until the range of coefficients of $\lambda \boldsymbol{d}$ was less than 10.)

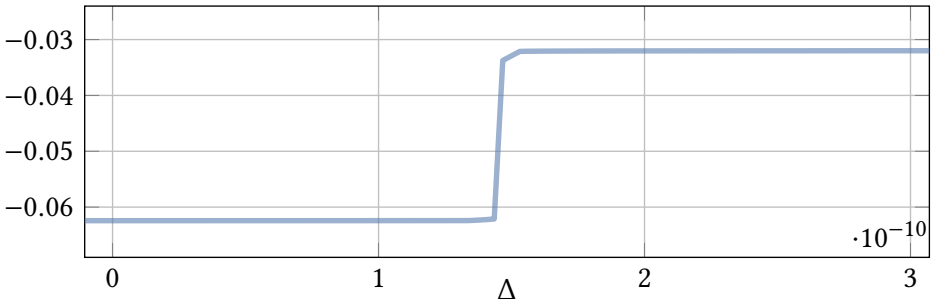


Figure 4.22: Projected gradient $\boldsymbol{d}^\intercal \boldsymbol{g}(\boldsymbol{u} + \lambda \boldsymbol{d})$ along the normalized Newton descent direction with step length $\lambda = 0.0217745227 + \Delta$.

5 Seamless Parametrization with Arbitrary Cones for Arbitrary Genus

5.1 Introduction

Computing global parametrizations of surfaces is a key operation in geometry processing. While in general only disk-like surfaces can be parametrized continuously in a (locally or globally) injective manner, surfaces of arbitrary topology can be dealt with by cutting them to disks. Across the cuts the parametrization will be discontinuous, but this is inevitable in general.

One can, however, require the parametric transitions across cuts to be from certain classes, in order to support specific applications like smooth surface approximation and quadrangulation. For instance, restricting to *similarity* transformations (rotation, translation, isotropic scaling) with a rotation by some multiple of $\pi/2$ yields global parametrizations ideal for T-spline constructions [Campen and Zorin 2017b]. Restricting further to *rigid* transformations with such discrete rotation angles yields parametrizations which (after quantization [Bommes et al. 2013b; Campen et al. 2015; Lyon et al. 2019]) are well suited for tasks like conforming quadrangulation, spline and subdivision fitting, seamless texturing, or constructing grids for solving PDEs on surfaces. We call such parametrizations *seamless* [Myles and Zorin 2012; Purnomo et al. 2004].

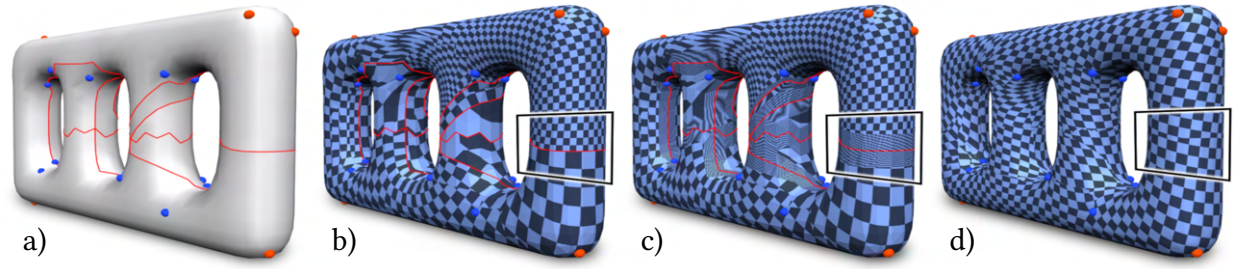


Figure 5.1: Method overview: a) Cut graph on a surface, consisting of handle loops, connectors, and one additional path. b) Conformal parametrization which maps the cut graph's branches to axis-aligned straight segments in the parametric domain and respects prescribed cone singularities (red and blue dots). This map is only *rotationally* seamless, i.e., rotational components of transitions across cuts are $k\pi/2$-rotations, $k \in \mathbb{Z}$, but scaling is arbitrary. c) This map modified by *map padding*; while locally highly distorted, it is actually seamless, there no longer is a scale jump. d) Result after optimization for low isometric distortion.

Seamless parametrizations can have *singularities*, points around which the total parametric angle is not 2π but some other integer multiple of $\pi/2$, i.e., the parametrization coordinate isolines do not locally form a regular grid. Equivalently, the metric induced by the parametrization has *cones*, points where the metric is not flat, its curvature not zero but some other integer multiple of $\pi/2$. Intuitively, in a quadrangulation induced by the parametrization, these singularities or cones correspond to extraordinary vertices, with valence different from 4.

As implied by the Gauss-Bonnet theorem, the total curvature of these cones is a topological invariant – i.e., such cones, which have a significant influence on the parametrization's quality and structure, cannot generally be avoided. Depending on the use case, they can be considered an impairment or features of special interest. In either case, having the ability to control (i.e., prescribe) them – where they are, how many there are, what curvature they have – is of obvious benefit. This motivates the problem we consider in this paper:

Compute a global seamless locally injective parametrization with prescribed cone positions and curvatures (respecting Gauss-Bonnet).

Known general approaches to this problem (cf. Sec. 2.3), e.g. those used as the initial, and most difficult, step in quadrangulation algorithms, rely on optimization formulations which are

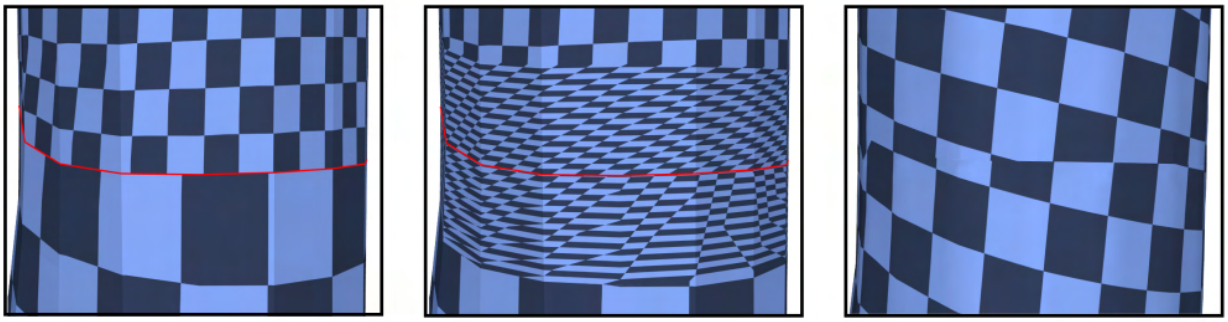


Figure 5.2: Zoom-ins of Figure 5.1. Left: cut-aligned conformal map. Middle: padded map, with high distortion, but seamless and locally injective. Right: map optimized for low isometric distortion.

non-convex and require feasible starting points to guarantee success; alternatively, convexified formulations may not yield a solution even if one exists. Thus, formally establishing existence of a solution and constructing a feasible starting point is an essential step in the general case. For the special case of genus 0, this task can be handled using existing conformal metric computation techniques [Springborn et al. 2008], [Luo 2004], [Gu et al. 2018a], [Campen and Zorin 2017b], cf. Sec. 5.3.3. Close to a general reliable solution to this problem is an approach by [Myles et al. 2014]: a valid global seamless injective parametrization is guaranteed, cone preservation is aimed for but not guaranteed – in a small fraction of cases unnecessary additional cones arise.

That these are truly unnecessary in almost all cases follows from the fact that the above task is actually feasible: the *existence* of such parametrizations follows from a theorem on meshes with prescribed extraordinary vertices [Jucovič and Trenkler 1973]. The proof is relatively complex and purely combinatorial. As a consequence, it does not easily translate into a practical parametrization construction.

We present a *constructive* proof for the existence of seamless surface parametrizations in this paper, that is conceptually simpler and translates to a parametrization algorithm. Precisely, we show:

Theorem 5.1. *Given a closed smooth surface M of genus g and an admissible set C of cones c_i, each given by a point $p_i \in M$ with a prescribed curvature value $\hat{\Theta}_i = (4 - k_i)\frac{\pi}{2}$, $k_i \in \mathbb{N}^{>1}$, there exists a*

global parametrization of M with cones C that has seamless transitions. We assume $k_i > 1$, as cones with curvature $3\frac{\pi}{2}$, corresponding to valence 1 vertices in a quadrangulation, are of low relevance in common applications; with some additional special case handling, our method could be extended to $k_i = 1$.

The terms used in the theorem are defined precisely in Sec. 5.2.

A set of cones $C = \{(p_i, k_i)\}$ is called *admissible* if it satisfies $\sum_i (1 - \frac{1}{4}k_i) = 2 - 2g$ (Gauss-Bonnet) and if $\boldsymbol{k} \neq (3, 5)$ (which is the single one notorious infeasible case [Jucovič and Trenkler 1973]).

Basic Idea

Instead of directly aiming for a seamless cone metric on a surface M,

1. we cut M open using a cut graph G, obtaining the cut surface M' consisting of one or more topological disks;

2. compute a cone metric on M' – *without* any seamlessness requirements, but with prescribed boundary curvature; specifically, we prescribe a *rectilinear* boundary, consisting of geodesically straight segments meeting at right angles;

3. modify this metric into a seamless one on M, yielding a seamless parametrization with the prescribed cones; exploiting the rectilinear boundary property, this modification is performed by *padding* the straight segments in the parametric domain with rectangles of suitably chosen size.

The metric in (2) is known to exist; e.g. a *conformal* metric with prescribed cones and boundary curvature (satisfying Gauss-Bonnet) on a disk always exists – in the smooth setting (cf. Sec. 5.2.2); the situation is more complicated in the discrete setting (cf. Sec. 5.2): questions concerning the exact conditions for existence of *discrete* conformal metrics with prescribed boundary curvature

as well as concerning the convergence of the existing algorithm that we leverage for this step, thus the injectivity of the derived map, remain open. We note that conformality is not essential here: any metric with prescribed boundary curvature could be used in that step.

Figures 5.1 and 5.2 show the outcome of these main steps. We refer to Appendix 5.7 for a comprehensive example illustrating these steps in a concrete, simple case.

Key Contributions

Our key technical contributions pertain to step (3) in the outline above:

- We propose a technique (*map padding*) to modify a non-seamless map into a seamless one.
- We prove that, for certain choices of cut graph combinatorics, this technique always succeeds.
- We describe an implementation of this construction for the discrete, piecewise linear case.

In other words, our approach is a problem reduction:

If one is able to compute a metric with prescribed cones and prescribed boundary curvature for *disk-topology surfaces*, this solves (by means of our technique) the more general problem of computing a global seamless parameterization with prescribed cones for *arbitrary-topology surfaces*.

Our algorithm can be used to obtain non-degenerate, locally injective, seamless parametrizations of arbitrary closed discrete surfaces (triangle meshes) with arbitrary cones, assuming the initial metric with prescribed boundary curvature can be obtained.

5.2 Seamless Parametrization Construction

First, we define the notion of a seamless parametrization as well as a weaker notion of a *rotationally seamless* parametrization we need as an intermediate step.

Suppose a smooth surface M is cut to a set of topological disks M_i^c by a cut graph G, i.e., a collection of smooth curves (*branches*) γ_j embeddeded in M meeting only at their endpoints (*nodes*). We call the resulting cut surface M^c; the boundary of M^c consists of curves γ_j^c (boundary curves). There is a canonical map $\pi : M^c \to M$, which is identity in the interior of M^c and maps exactly two boundary curves γ_j^c to each branch γ_j on M. Pairs of boundary curves mapping to the same branch γ_j are called *mates*, and boundary points where curves γ_j^c meet are called *joints*. The image of any joint under π is a node. Pairs of non-joint points $p, q \in \partial M^c$ with $\pi(p) = \pi(q)$ are called *mated points*. For a boundary point p, let $t_p \in T_p M^c$ denote a unit vector that is tangent to the boundary ∂M^c at p.

Definition 5.2 (Rotationally Seamless Parametrization)**.** A continuous, locally injective map $F : M^c \to \mathbb{R}^2$ is called *rotationally seamless* parametrization of M, if for any pair p, q of mated points, the images $dF_p(t_p)$ and $dF_q(t_q)$ of boundary tangents are related by a similarity transformation T_{pq}, i.e., $T_{pq} \circ dF_p(t_p) = dF_q(t_q)$, with a rotation angle that is a multiple of $\pi/2$ and constant per branch.

Such a rotationally seamless parametrization does, in general, have a (pointwise) *scale jump* across the cut (cf. Fig. 5.3 left) – unless the similarity T_{pq} is actually just a rotation everywhere:

Definition 5.3 (Seamless Parametrization)**.** A map $F : M^c \to \mathbb{R}^2$ is called a *seamless* parametrization of M, if it is rotationally seamless and for each pair of mated points p, q the transition T_{pq} is

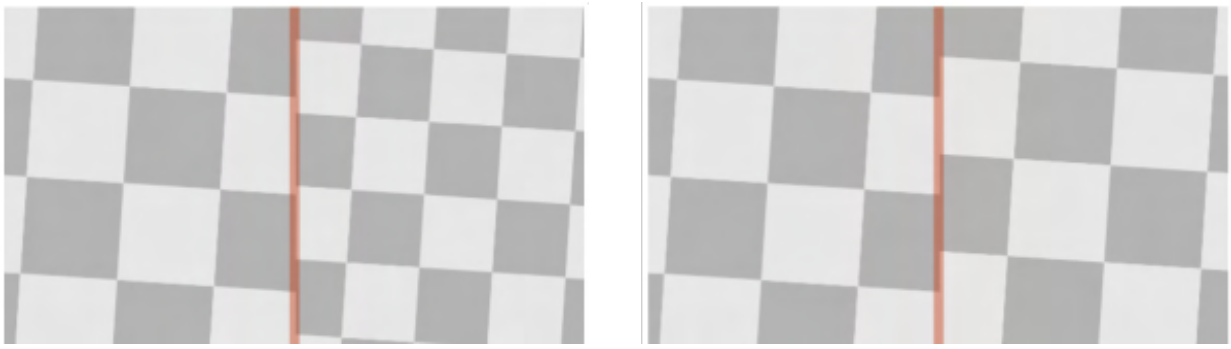


Figure 5.3: Visualization of a parametrization on a surface near a cut branch (red). Left: rotationally seamless. Right: seamless.

rigid, i.e., it is a rotation with a rotation angle that is a multiple of $\pi/2$.

Notice that seamlessness implies that the images $F(\gamma_j^c)$ and $F(\gamma_k^c)$ of mates γ_j^c and γ_k^c are congruent.

A seamless parametrization induces a metric on the surface M which is flat except at the nodes, where it may be singular; it may have a *cone*. We say that a seamless parametrization has a cone with angle α at a node p, if the sum of parametric angles at all joints q in M^c with $\pi(q) = p$ is equal to α. This cone has curvature $\Theta = 2\pi - \alpha$.

By contrast, the notion of a rotationally seamless parametrization is weaker: due to the scale jump, it does not induce a metric on M.

Overall Approach We first construct a parametrization F that is rotationally seamless, using a specific type of (conformal) maps: maps with rectilinear boundary, i.e., with the image of the boundary of the cut surface consisting of straight segments meeting at right angles. Then this parametrization is modified near the boundary to make the scale jump vanish so as to make it into a seamless parametrization F^s. This is done using a process we call *map padding*. The key to our construction is cutting the surface into *two* (in special cases three or four) topological disks, using cut graphs with a particular structure. This is important for our method of converting rotationally seamless parametrizations into seamless parametrizations.

5.2.1 Cutting to Disk(s)

We construct the required cut graph G in two steps, first cutting the surface M into a set of topological disks M_i'. Their disjoint union is denoted M'. Typically we use two disks, with some exceptions for special genus 2 cases. M' contains all cones in its interior. The final cut surface M^c is obtained by adding branches passing through all cones to this cut graph, such that no M_i' is split into multiple components. This second step is explained in Sec. 5.2.3.

For the first step, we consider a particular type of cut graphs that only have nodes of degree 4 and 3. Pairs of cyclically sequential branches around nodes form sectors: four at degree 4 nodes, three at degree 3 nodes. At degree 4 nodes, all four sectors are marked as *corners* (cf. Fig. 5.5). At

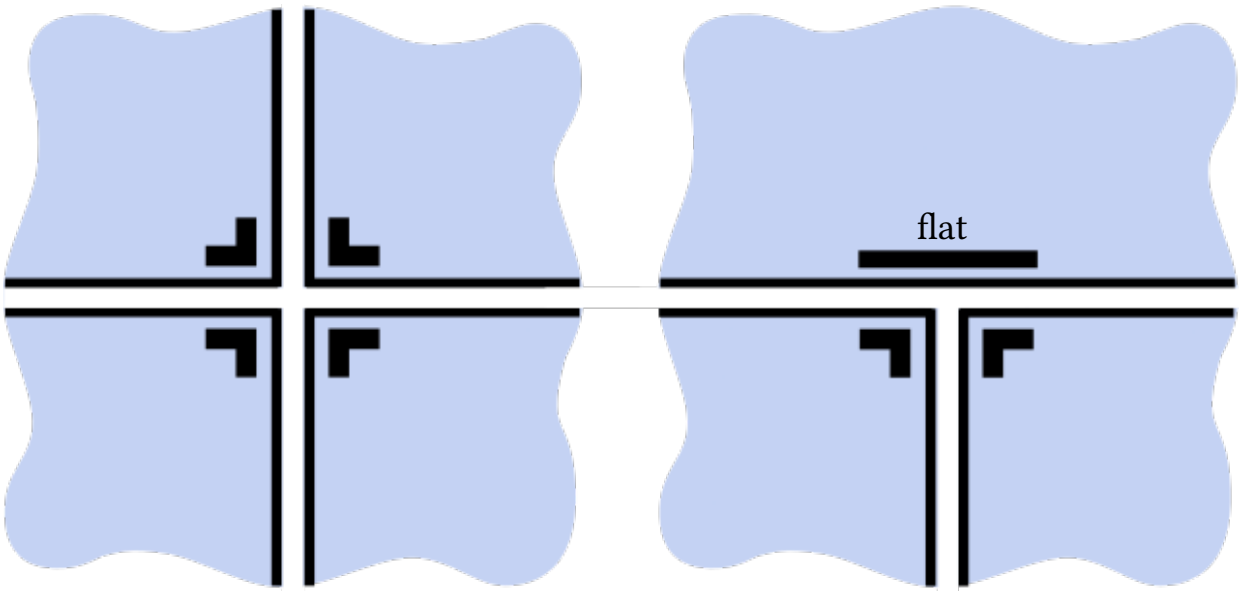


Figure 5.4: Two different type of nodes, degree 4 (left) and degree 3 (right), are shown.

degree 3 nodes, two sectors are marked as corners, the third one is referred to as *flat*. We refer to degree 3 nodes as *T-nodes*.

We denote the boundary curves of M' by γ'_j. Any pair of sequential boundary curves of M' corresponds to a corner or a flat joint. As we will require boundary curves to be straight and corners to have angles $\pi/2$ under a certain metric in the following, the number m_i of corners on the boundary of each connected component M'_i needs to match the total prescribed cone curvature in the interior of M'_i as per Gauss-Bonnet, i.e.,

$$m_i \frac{\pi}{2} + \sum_{(p_j, \hat{\Theta}_j) \in C'_i} \hat{\Theta}_j = 2\pi, \tag{5.1}$$

where $C'_i \subseteq C$ is the subset of cones prescribed within M'_i. Note that this is equivalent to $m_i = 4 + \sum_{(p_j, \hat{\Theta}_j) \in C'_i} (k_j - 4)$.

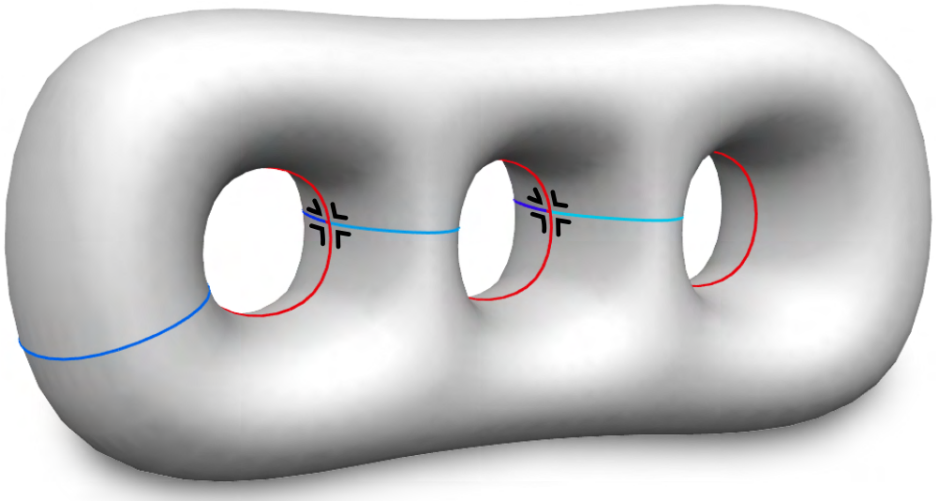


Figure 5.5: Degree 4 cut graph on a surface of genus $g = 3$. This cut graph has 10 branches and 5 degree 4 nodes, thus 20 corners (marked black). The cut graph consists of loops (red) and connectors (shades of blue) (cf. Sec. 5.3.1)

Definition 5.4 (Admissible Cut Graph)**.** A cut graph with marked corners is *admissible*, if

- all branches are embedded smooth curves meeting transversally at nodes of degree 3 or 4, and not passing through cones;
- the cut graph partitions the surface into disk-topology components;
- the number of corners of each component satisfies Eq. (5.1);
- if a boundary curve is involved in a flat sector, its mate is not.

5.2.2 Cone Metric with Rectilinear Boundary

Corners partition the boundary $\partial M'$ into *segments*. Note that a segment may contain flat joints, and, as a consequence, consist of several boundary curves γ'_j (*complex segment*). (All cut graphs we will be working with contain at most two T-nodes, thus two flat joints, i.e., almost all segments are simple segments.)

We now require a cone metric on M' which has a *rectilinear* boundary: under such a metric, segments are geodesically straight (i.e., zero geodesic boundary curvature along $\partial M'$ in the interior of segments), and sequential segments form right inner angles of $\pi/2$.

Proposition 4. *On a cut surface M', obtained from a smooth surface M by cutting it along an admissible cut graph G, there is a cone metric with rectilinear boundary and prescribed admissible cones $C = \{(p_i, \hat{\Theta}_i)\}$.*

A proof is given in Appendix 5.9. In particular, a *conformal* cone metric with these properties exists; conformality, however, is not essential in the following. This cone metric on M' does not, in general, define a metric on M, as the lengths defined by the metric along the two sides of the cut graph's branches may disagree, cf. Fig. 5.2 left.

Note that the metric angle on M around points on G is 2π everywhere: points in the interior of branches are surrounded by two sectors with angles $\pi + \pi$, node points by three sectors with angles $\pi + 2\frac{\pi}{2}$ or four sectors with angles $4\frac{\pi}{2}$. As we preserve these angles in the following, this implies that no spurious cones emerge.

5.2.3 Metric to Rotationally Seamless Parametrization

The cone metric with rectilinear boundary is flat away from the cones on M'. We now extend the cut graph G by a set of trees T_i, yielding the extended cut graph $G^T = G \cup T$, where T is

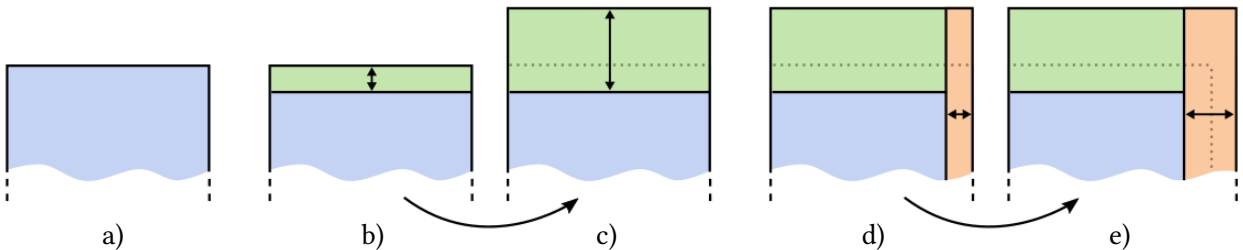


Figure 5.6: a) Generic local view of the boundary of map $F(M^c)$, with straight segments and right-angle corners. b) A rectangular strip along a segment is marked. c) The strip is stretched outwards, effectively increasing the length of the two adjacent segments left and right of the central segment. d) This padding operation can be applied in sequence to further segments.

the union of trees T_i. The tree T_i is rooted on $\partial M_i'$ at a single non-joint point, and its branches connect all cones prescribed within M_i'. Let M_i^c be the surface obtained by cutting M_i' along T_i. The segment of $\partial M_i'$ split by the root point of T_i is still considered one logical segment. Note that this cutting of M' to M^c by T introduces additional boundary curves; we do *not* refer to these as segments, and they are, in general, not straight under the cone metric. The boundary curves of M^c are denoted γ_j' if they map to branches of G, or γ_j^T if they map to branches of T. The above constructed cone metric is flat in the interior of M_i^c (as the cones lie on ∂M^c), and defines (via integration) a map $F_i : M_i^c \to \mathbb{R}^2$. It is unique up to a rigid transformation. We choose this transformation so that all segment images are axis-aligned in $\mathbb{R}^2$. This is possible because they (due to rectilinearity) are all straight and meet at right angles. Note that images of boundary curves γ_j^T are, in contrast to segments (consisting of boundary curves γ_j'), neither axis-aligned nor straight in general, cf. Fig. 5.19 in Appendix 5.7. Together, these maps F_i define a global parametrization F of M.

Proposition 5. *The map F is a rotationally seamless parametrization of M (but not, in general, seamless – except on T).*

Proof. Due to all segment images being axis-aligned, the angle between the images of any two mated boundary curves γ_j', γ_k' is some multiple of $\pi/2$, constant per branch. The images of any two mated boundary curves γ_j^T, γ_k^T are congruent (in particular, similar) as the metric is flat on T by construction. The rotation between them is a multiple of $\pi/2$ because the prescribed angles at cones are multiples of $\pi/2$ (cf., e.g., [Springborn et al. 2008]). Hence, F is seamless on T but, in general, only rotationally seamless on G. □

Visualization For purposes of illustration, we would like to visualize the image $F(M^c)$. Due to global overlaps implied by negative curvature cones, this is not an easy task. However, locally, near the cut graph G, $F(M^c)$ always looks like in Fig. 5.6a – because the boundary consists exclusively of straight segments meeting at right-angle corners. (The only exception being the

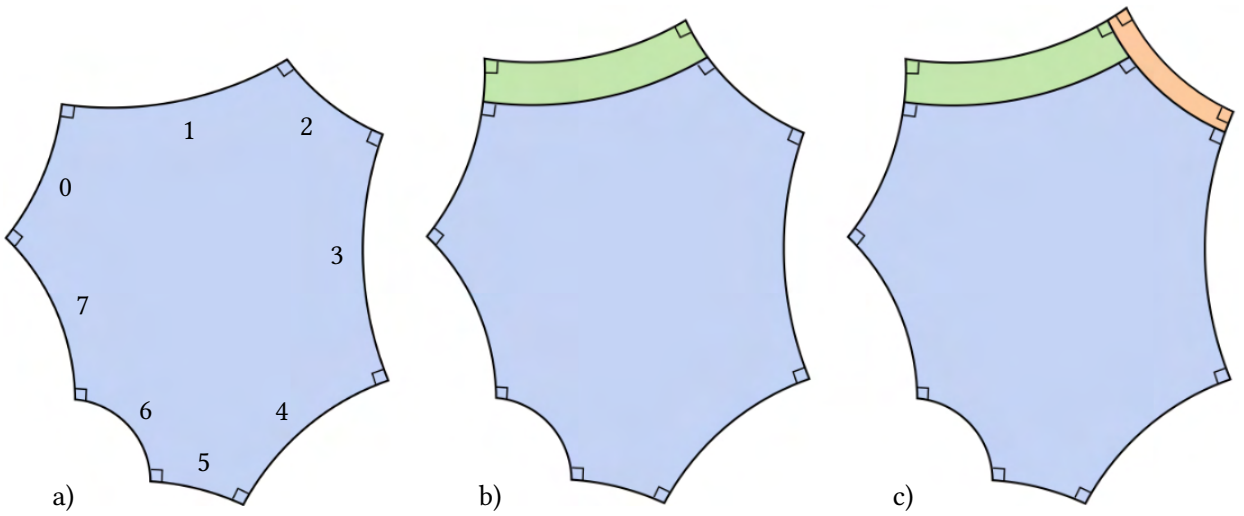


Figure 5.7: a) Global visualization (without cuts to cones) of the rectilinear map, where straight segments appear as curved arcs (as explained in Sec. 5.2.3). b) Padding (analogous to Fig. 5.6) of segment 1, increasing the lengths of segments 0 and 2. c) Padding of segment 2, increasing the lengths of segments 1 and 3. This can be continued to adjust all segments' lengths.

one boundary curve per M_i where the tree T_i is rooted.) We use this type of illustration when a local view is sufficient. An alternative is to flatten the surface globally, without cutting to the cones, instead (for visualization purposes) pushing the curvature of the cones evenly onto the boundary $\partial M'$. This leads to a flattening of M_i' as shown in Fig. 5.7a, where straight boundary segments appear as curved arcs (and cones are not visible). This makes it possible to visualize the complete rectilinear boundary without cuts or overlaps.

5.2.4 Seamless Parametrization by Padding

The rotationally seamless map F differs from a seamless map in two respects: the images of mated segments may have different lengths, which implies a scale jump; but even if they are of equal length, this only implies that the scale is equal on average, rather than pointwise along the corresponding branch.

We thus modify F by composing it with two types of local segment-wise maps:

- a *stretch* map g_j which effects a change of the lengths of segment images,

- a *shift* map r_j which subsequently equidistributes scale along a segment.

We iteratively apply these operations segment by segment. Note that these are applied only to *segments*; across the additional cuts T the map F is already seamless, cf. Prop. 5.

STRETCH For a boundary segment s_j, we consider a thin strip S_j on M^c which runs along the entire segment and maps to a rectangular region R_j via F. This is illustrated in Fig. 5.6b and Fig. 5.8.

More formally, the strips are defined as follows. The restriction of F to the segment s_j (which maps s_j to a straight segment in the plane) is bijective, and so is the restriction F_j to a sufficiently small neighborhood $\Omega_j \subset M^c$ of s_j. We choose the rectangle R_j within $F(\Omega_j)$ such that it includes $F(s_j)$ but no cone points and no joints except the ones on s_j. The thin strip S_j on M^c is then defined as $S_j = F_j^{-1}(R_j)$, as shown in Fig. 5.8.

Outside of S_j we preserve the map, but within S_j we modify it by a one-dimensional scale map g_j such that S_j is mapped onto a larger rectangle $R'_j \supseteq R_j$ whose width (orthogonal to the segment s_j) is increased by a *padding width* w_j. This is illustrated in Fig. 5.6c and Fig. 5.8. Effectively, the domain is locally *padded* by an additional rectangular region $R'_j \backslash R_j$ of width w_j along the image of s_j, cf. Fig. 5.7.

The situation is slightly different at the one segment s_j per component where T is rooted (cf. Sec. 5.2.3): it is separated into two parts by T (cf. Fig. 5.21 left). Both parts can, however, be

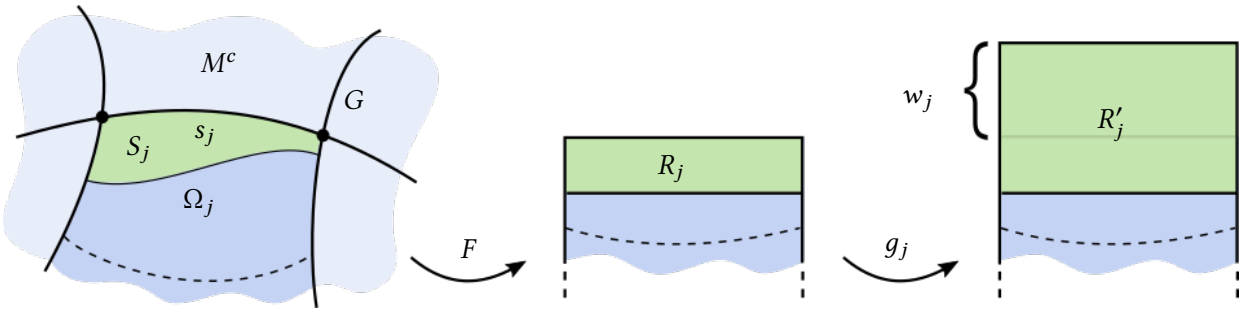


Figure 5.8: Illustration of strip definition and stretch map applied to perform padding of a segment s_j by padding width w_j, cf. Sec. 5.2.4.

handled separately using the same technique, as detailed in Appendix 5.10.

We define the padded map F^p iteratively, iterating over the (arbitrarily ordered) strips S_j, $j = 1, \ldots, n$, of each connected component of M^c. $F^{p,0}$ coincides with F, and $F^{p,m+1}$ differs from $F^{p,m}$ only on R_{m+1}, where it is defined as $g_{m+1} \circ F^{p,m}|_{R_{m+1}}$, where g_{m+1} is the above scaling transformation (detailed in Appendix 5.10). $F^p = F^{p,n}$.

Shift In the padded map F^p, we consider for each segment a strip S_j^p (now defined based on F^p) and modify the map within this strip by a map r_j with the following properties:

- its restriction to a simple segment s_j reparametrizes s_j to constant speed; for complex segments: constant speed per contained boundary curve, cf. Appendix 5.10,
- the map is equal to identity on the rest of the strip's boundary,
- it is continuous and bijective.

We define the shifted map F^s iteratively, again iterating over the strips S_j^p of each connected component of M^c. $F^{s,0}$ coincides with F^p, and $F^{s,m+1}$ differs from $F^{s,m}$ only on $R_{m+1}^p = F^{s,m}(S_{m+1}^p)$, where it is defined as $r_{m+1} \circ F^{s,m}|_{R_{m+1}^p}$, where r_{m+1} is the above shift map (detailed in Appendix 5.10). Finally, $F^s = F^{s,n}$.

Proposition 6. *F^s is a rotationally seamless parametrization of M with the same cones as F, and not only the angle but also the scale jump is constant per branch of the cut.*

A proof is given in Appendix 5.10. The choice of padding widths determines the lengths of segments under F^p and, as they are preserved by the shift maps, under F^s. In the following Sec. 5.2.5 we detail how *equalizing* padding widths can be found:

Definition 5.5 (Equalizing Padding Widths)**.** A set of padding widths leading to all pairs of mates being of equal length under F^s is called *equalizing padding widths*.

Proposition 7. *If F^s is constructed using equalizing padding widths, it is a seamless parametrization of M.*

Proof. The constant scale jump per branch (Prop. 6), together with the equal lengths of mated boundary curve images implies a scale jump of zero per branch, i.e., rigid transitions. □

5.2.5 Length Equalization

When a boundary segment is padded, the lengths of the two adjacent segments' images change. To make this precise, let ℓ_i be the length of segment s_i before any padding is performed, and w_i be the amount of padding applied to s_i. The length ℓ_i' of s_i after each segment was padded according to values $\boldsymbol{w} = (w_0, w_1, w_2, \ldots)$ is

$$\ell_i' = w_{\text{prev}(i)} + \ell_i + w_{\text{next}(i)}, \tag{5.2}$$

where $\text{prev}(i)$ and $\text{next}(i)$ are the two segments adjacent to s_i, preceding and following it in cyclic order along $\partial M'$, cf. Fig. 5.9.

Our goal is to find an equalizing assignment of padding width variables $\boldsymbol{w}$ such that the lengths $\boldsymbol{\ell}' = \{\ell_0', \ell_1', \ell_2', \ldots\}$ after padding are equal for each pair (s_i, s_j) of mated segments, i.e., $\ell_i' = \ell_j'$. This leads to *length equalization* equations

$$w_{\text{prev}(i)} + w_{\text{next}(i)} - w_{\text{prev}(j)} - w_{\text{next}(j)} = \ell_j - \ell_i. \tag{5.3}$$

However, only if all cut graph nodes are of degree 4, all segments are simple, and mated in pairs as a consequence. In the presence of T-nodes (which imply complex segments) the situation is a little different: a complex segment s_i consists of multiple boundary curves; their mate curves, however, form simple segments due to the last property of Def. 5.4. Hence, generally, a (simple or complex) segment s_i is mated with a sequence $J_i = (s_j, s_k, \ldots)$ of one or more simple segments.

Length equalization equations then take this more general form:

$$\ell'_i = \sum_{j \in J_i} \ell'_j \tag{5.4}$$

which expands to

$$w_{\text{prev}(i)} + w_{\text{next}(i)} - \sum_{j \in J_i} \left(w_{\text{prev}(j)} + w_{\text{next}(j)} \right) = \sum_{j \in J_i} \ell_j - \ell_i. \tag{5.5}$$

These equations form a globally interdependent equation system:

$$A\boldsymbol{w} = \boldsymbol{b}, \quad w_i \geq 0 \, \forall i. \tag{5.6}$$

Note the non-negativity condition; it ensures that the padding operation actually stretches ($w > 0$) rather than squeezes ($w < 0$) the strips along segments. Allowing squeezing would require an upper-bound constraint.

This system needs to be solved to achieve length equalization of mated segments, enabling seamlessness. Unfortunately, it is not generally feasible – it may have no non-negative solution or even no solution at all. Notice that the system matrix structure is entirely determined by the cut graph's combinatorics, leading to the following definition.

Definition 5.6 (Equalizable Cut Graph)**.** An admissible cut graph for which equalization system (5.6) is feasible for arbitrary $\boldsymbol{b}$, is called *equalizable*.

Our key result, accompanying the proposed map padding technique, is a proof showing that there is an equalizable cut graph for any genus and any admissible set of cones, as well as an efficient algorithm to construct such cut graphs (cf. Sec. 5.3 and Appendix 5.8).

5.3 Equalizable Cut Graphs

We prove the following proposition:

Proposition 8. *For any genus g and any admissible prescription of cones C, there is an equalizable cut graph, i.e., we can always obtain equalizing padding widths that enable a seamless parametrization.*

The foundation of our construction of equalizible cut graphs is a graph we call a *hole chain*. Variations thereof, depending on the surface's genus and the cone configuration, then yield equalizable cutgraphs. Cut graphs covering all cases (arbitrary genus, arbitrary admissible cones) are defined in this section. Their equalizability is shown in Appendix 5.8.

Together with Prop. 4 (rectilinear cone metric), Prop. 5 (rotationally seamless map), and Prop. 7 (seamless modification given equalizing padding widths), this Proposition (equalizable cutgraphs) **concludes the constructive proof of the main theorem 5.1.**

5.3.1 Hole Chain

Given a closed surface M of genus $g > 0$, we cut it along g non-intersecting non-homotopic non-separating smooth loops α_i. This yields a topological sphere M° with $2g$ holes. Note that each loop corresponds to two holes, which are called *partners*. Let the holes be numbered from 0 to $2g-1$, and denoted h_i, in such a way that h_0 and h_{2g-1} (called *terminals*) are partners.

Let $\pi : M^\circ \to M$ be the canonical map from M° to M, taking h_i and its partner h_j to their corresponding loop α: $\pi(h_i) = \pi(h_j) = \alpha$.

On each hole h_i, pick two distinct points q_i and q_i', such that they are identified across partners on M, i.e., for partners h_i, h_j we have $\pi(q_i) = \pi(q_j)$ and $\pi(q_i') = \pi(q_j')$. For each $0 \leq i < 2g-1$ we further cut M° along a smooth non-intersecting path between holes h_i to h_{i+1}, starting transversally at point q_i' and ending transversally at point q_{i+1}. These paths are called *connectors*.

Note that after each such cut the surface remains a topological sphere with holes (decreasing in number), thus, it remains path-connected; therefore these connectors always exist. Loops and connectors together form a cut graph we call *hole chain*, as depicted abstractly in Fig. 5.10 and concretely in Fig. 5.5, which yields the surface M', a sphere with one hole, i.e., a disk. Loops and connectors are assumed not to intersect any prescribed cone.

Proposition 9. *The hole-chain cut graph for any genus $g > 0$ is admissible.*

Proof. As the connectors' endpoints q_i, q_j are identified in pairs on M across partners h_i, h_j, each resulting cut graph node (at point $\pi(q_i) = \pi(q_j)$ on M) is of degree 4 (cf. Fig. 5.5). All branches are smooth curves meeting transversally at their endpoints and not crossing cones by construction. The surface is cut to a single component with disk topology. As the set of cones C is admissible, we have $\sum_j \Theta_j = 2\pi(2-2g)$; the hole chain cut graph has $2g-1$ nodes with 4 corners each, i.e., $8g-4$ corners. Thus Eq. (5.1) is satisfied. □

Odd-Couple Condition We impose one condition (besides terminals being partners) on the way the numbering of holes is chosen: there needs to be at least one odd couple, i.e., two partner holes which have an odd number of holes between them in the chain, i.e., there is an i and an integer k such that h_i and h_{i+2k} are partners. This will be expected in the proof of equalizability. Note that this is impossible if there are just four or less holes, thus instead special case variations of the hole chain are used for genus 1 (two holes) and genus 2 (four holes) cases, as detailed in Sec. 5.3.3.

Definition 5.7 (Fourfold Cones)**.** A set of cones $C = \{(c_i, k_i)\}$ with k_i divisible by 4 for each i is called *fourfold*.

While the above hole chain cut graph is not equalizable in general, it permits equalization for *specific* righthand sides $\boldsymbol{b}$:

Proposition 10. *For any genus, a fourfold cone prescription implies a righthand side* $\boldsymbol{b}$ *for which problem* (5.6) *is feasible.*

A proof is given in Appendix 5.8. For the general, non-fourfold case, variations of the hole chain are used. We will describe the necessary changes next.

5.3.2 General Case (Genus 3+)

In case the cone prescription is not fourfold, i.e., there is at least one $k_i \mod 4 \neq 0$, we extend the hole chain cut graph by one extra path (cf. Fig. 5.11) – which makes it equalizable.

Definition 5.8 (Valid Extra Path)**.** A simple path is called a valid extra path for a hole chain cut graph if

- it does not pass through a cone,
- only its endpoints are contained in the hole chain cut graph,
- at least one endpoint is on a hole of the hole chain,
- no endpoint is coincident with a node of the hole chain.

Notice that this extra path forms two additional nodes, both of degree 3, i.e., T-nodes (or, if coincident, one of degree 4) at its endpoints. At each we mark as corners the two sectors directly adjacent to the extra path. Hence the total number of corners increases from $8g-4$ to $8g$. At the same time, the extended hole chain cut graph cuts the surface into two components, M_0' and M_1', each with disk-topology, with numbers of corners m_0, m_1. In particular, we have $m_0 + m_1 = 8g$.

In order for the extended hole chain to remain admissible, we need to ensure that Eq. (5.1) is satisfied, i.e., the number m_i of corners per component M_i' needs to match the total curvature of cones C_i' prescribed within the component. Note that this is satisfied for M_0' if and only if it is satisfied for M_1'. Also, if the endpoints lie on two mated segments, we choose them as mated

points, to create a degree 4 node rather than two opposite T-nodes. We furthermore require that the numbers m_0, m_1 of corners are not divisible by 4. This will be expected in the proof of equalizability.

Definition 5.9 (Admissible Extra Path)**.** A valid extra path that yields corner numbers m_0, m_1 not divisible by 4 and satisfying Eq. (5.1) is called admissible.

Proposition 11. *An admissible extra path exists for any non-fourfold cone prescription and any genus $g \geq 3$.*

Proof. Pick one prescribed cone c_i with $k_i \bmod 4 \neq 0$. Let β be a simple path from a non-corner point q on the hole h_0 to c_i, not containing any other cone. Let γ be a path that starts at q, runs (arbitrarily close) along one side of β, then around c_i, then back along the other side of β, and ultimately (arbitrarily close) along the cut G until it has passed $k_i - 2$ corners. It then connects to a point q' on the segment it reached. If γ is chosen sufficiently close to β and G, it is an example of a valid extra path: the region that contains c_i contains no other cone and it has k_i corners (the $k_i - 2$ corners passed along G plus the two corners formed by the extra path itself with G at q and q'). Also, γ is connected to a hole, namely h_0. □

The equalizability of the hole chain cut graph with an odd-couple extended by an admissible extra path is proven in Appendix. 5.8.

5.3.3 Special Cases (Genus 0, 1, 2)

Genus 0 Case

In the case of a topological sphere, our method formally is applicable, but does not actually contribute anything: the cut graph is empty; there are no cuts across which the cone metric could be non-seamless, thus no padding is required. The existence of conformal metrics with prescribed cones on the topological sphere M is well-known [Troyanov 1991].

Genus 1 Case

For genus 1 surfaces, we (similar to the general case) add one extra path, but deviate slightly from the general hole chain pattern in terms of identification of connector endpoints. The cut graph pattern is depicted in Fig. 5.12. Notice that the surface is split into two components, with 2 and 6 corners. It is easy to see that for any admissible prescription of cones on a genus 1 surface, one either has no cones at all (in this case the basic hole chain cut graph is sufficient, cf. Prop. 10) or one has, among the prescribed cones, one or more cones whose curvature sums up to π (due to the Gauss-Bonnet theorem, there are cones of positive and negative curvature, and the case of a single positive cone of curvature $\pi/2$ is the one non-admissible case, cf. Sec. 5.1). The surface bi-partition by this cut graph pattern is thus compatible with any non-empty cone prescription, i.e., the paths can be chosen in an admissible way on M.

The equalization equation system of this pattern is easily checked explicitly for non-negative feasibility (cf. Appendix 5.8.2).

Genus 2 Case

For genus 2 surfaces, we also need to deviate from the general case. In contrast to the genus 1 case, where we could assume that a subset of prescribed cones always have curvatures summing up to one specific value, five cases need to be distinguished: there is a subset of prescribed cones with curvatures summing to π, $\pi/2$, $-\pi/2$, $-\pi$, or $-3\pi/2$ (compatible with regions with 2, 3, 5, 6, and 7 corners, respectively). This list is exhaustive because the total sum of cone curvatures is -4π on a genus 2 surface, and not all cones have curvatures that are multiples of 2π (as this case was handled already in Prop. 10); as a consequence, at least one of these five values has to appear as a subsum.

Depending on which curvature sum subset is available in a given set of prescribed cones, the cut graph pattern needs to be chosen compatibly. For the case that a cone subset with curvature

sum $-\pi/2$ is available (as in most practical scenarios), the pattern depicted in Fig. 5.13 can be used; for the remaining patterns refer to Appendix 5.8.3. Notice that this pattern is a variation of the basic hole chain: two connectors are required to cross. This partitions the surface into a 5-corner region (compatible with a curvature $-\pi/2$ subset) and a 11-corner region (compatible with the remaining cones).

The equation systems corresponding to these patterns are easily checked explicitly for non-negative feasibility.

5.4 Implementation

We now describe how our algorithm can be implemented for discrete surfaces. In this section, M denotes a closed triangle mesh of arbitrary genus g. Cones are prescribed at vertices of M; such vertices are called *cone vertices*. We first focus on the case $g \geq 3$, then on the minor deviations required for cases $g = 1$ and $g = 2$.

General Overview The main steps of the algorithm are:

1. Construct g non-contractible loops and cut M along these loops (Sec. 5.4.1).

2. Connect all holes using shortest paths, selecting the connection pattern based on the given genus and cones. Where necessary, add one extra path; then cut the mesh (Sec. 5.4.1).

3. Set target angles at cone and corner vertices. Compute the corresponding discrete conformal metric with prescribed curvature (Sec. 5.4.2).

4. Number all cut segments and set up the system matrix A and righthand side $\boldsymbol{b}$ accordingly (Sec. 5.4.3).

5. Compute a solution to the linear system $A\boldsymbol{w} = \boldsymbol{b}$; add a constant shift to yield a solution $\boldsymbol{w} \geq 0$ (Sec. 5.4.3).

6. Extend the cut graph to include all cones; lay out the mesh in the plane according to the metric (Sec. 5.4.4).

7. Perform padding according to padding widths $\boldsymbol{w}$ (Sec. 5.4.5).

The output of the algorithm is a seamless parametrization with the prescribed cones. While the parametric distortion initially is high near the cuts, this parametrization provides the *feasible starting point* required by techniques for injectivity-preserving parametrization distortion optimization (cf. Sec. 5.4.6) as well as quadrangulation methods based on quantization of seamless parametrizations.

5.4.1 Cut Graph

On M, one option to obtain g non-intersecting non-contractible loops is via handle/tunnel loop algorithms [Dey et al. 2013], modified to avoid cone vertices. A simpler robust approach is to iteratively cut the mesh g times, each time along an arbitrary non-contractible loop not containing a cone or a boundary vertex, obtained using the tree-cotree algorithm [Erickson and Whittlesey 2005]. To ultimately yield a cut graph that is not unnecessarily convoluted, it is advisable to pick a short loop each time.

We construct the connectors between the $2g$ holes as shortest paths, not containing cone, boundary, or other paths' vertices, using Dijkstra's algorithm. A natural ordering of the holes can be determined using a Hamiltonian path algorithm; this order needs to be adjusted slightly before connector construction, to ensure the paired-terminals and odd-couple conditions are satisfied (cf. Sec. 5.3.1).

Mesh Refinement We work with discrete paths/loops, following the edges of M. For the algorithm to be robust regardless of the mesh structure, after each construction of a path, we split each mesh edge that is not on a path if its two vertices both are either on the boundary, on a path, or on a cone. This ensures that the mesh, cut by the paths, remains path-connected with respect to the discrete edge paths, avoiding boundary, path, and cone vertices.

Extra Cut The extra cut path (needed for the general genus $g \geq 3$ case) is constructed as a shortest path as well. However, we need to employ a cone-aware variant of Dijkstra's algorithm in order to ensure cone/corner compatibility, cf. Eq. (5.1).

To this end, to each directed dual edge e of the mesh M' we assign a value ρ_e (with $\rho_e = -\rho_{\bar{e}}$ for oppositely directed dual edges $e, \bar{e}$) such that the sum of these values clockwise around a single cone vertex c_i is $\hat{\Theta}_i$, and around non-cone vertices zero. Such an assignment can, for instance, be obtained using a spanning tree of the cones (Fig. 5.14), rooted at the boundary $\partial M'$: initialize ρ at all leaves of the tree and propagate the values towards the root, summing values where branches meet.

Now for an arbitrary simple closed clockwise dual edge path γ, we have the following important property: $\sum_{e \in \gamma} \rho_e = \sum_{p_i \in \Gamma} \hat{\Theta}_i$, where Γ is the set of all vertices enclosed by γ [Crane et al. 2010]. We call this sum $\sum_{e \in \gamma} \rho_e$, with a slight abuse of terminology also for non-closed paths, *(partial) holonomy*. For a closed path, the sum $\sum_{e \in \gamma} \rho_e$ along a closed path tells us what total cone curvature is contained in the region enclosed by the path.

We then employ Dijkstra's algorithm, starting from a hole segment on $\partial M'$, and keep track of the partial holonomy values along the way. Whenever the front propagation in Dijkstra's algorithm reaches $\partial M'$ again, we tentatively close the loop by walking back to the starting point clockwise along $\partial M'$, counting passed corners on the way, and checking whether the total holonomy matches the number of corners, cf. Sec. 5.2.1, Eq. (5.1). If it matches, the path is accepted and added as extra cut. An example is shown in Fig. 5.14.

A modification is needed to the standard algorithm, though: Dijk-stra's algorithm keeps track of, for each vertex, the shortest path back to the starting point – regardless of partial holonomy $\sum \rho$. So while there are shortest paths of different partial holonomy back to the starting point, Dijkstra's algorithm discards all but the shortest one. We, instead, keep track of the shortest path *per vertex per holonomy value*. Otherwise paths that could end up having a suitable holonomy in the end, may be discarded early. We therefore perform Dijkstra's algorithm not on M', but on a branched covering of M' [Kälberer et al. 2007], with sheets glued according to ρ. In practice, this means that each triangle stores separate distance information per partial holonomy value, indexed by $\sum \rho$ of incoming fronts.

We also need to ensure that γ is simple on M'. While the holonomy-aware version of Dijkstra's algorithm yields a simple path on the covering, its projection to M' may be self-intersecting. Before advancing the front to the next vertex, we always check whether this vertex is already contained in the predecessor path to prevent such self-intersections. While with this latter modification it is no longer guaranteed that a path is always found, one can always fall back to an explicit path construction following the existence proof in Sec. 5.3.2; we have never encountered a case where this was necessary.

Special Cases The connectors of the special cut graph patterns employed for genus 1 and genus 2 surfaces are realized using shortest paths as well. These are constructed incrementally between endpoints chosen on the holes, and cross points chosen on other connectors where necessary. For those paths that split the surface into disjoint components, again the above cone-aware shortest path algorithm is employed to ensure cone/corner compatibility.

5.4.2 Conformal Map

After cutting M using the cut graph G to obtain M', for each component of M' (typically one or two, except for some genus 2 cone configurations) we need to obtain a cone metric with rec-

tilinear boundary. This is the one part of the implementation, where achieving robustness in the discrete case is a challenge, even though in the smooth case (cf. Sec. 5.2.2) things are rather straightforward.

In contrast to the continuous case, questions of existence of (some notion of) *discrete* conformal metrics with prescribed cones and boundary curvature are not fully settled. For cases without boundary, recent results have brought theoretical insights [Luo 2004; Gu et al. 2013,0; Springborn 2017] and provide an algorithmic foundation but the boundary case requires further work on the theory side.

We use a modification of the conformal mapping algorithm for meshes without boundary described in [Campen and Zorin 2017b]; it combines the elegant variational formulation of [Springborn et al. 2008] with on-the-fly mesh modifications (edge flips, following [Luo 2004]). For the genus 0 case, this algorithm can be used directly (cf. Sec. 5.3.3) to produce a seamless parametrization. With a minor extension, we additionally prescribe geodesic boundary curvature, using the holonomy angle constraints offered by this method, applied for each boundary vertex's triangle fan. The angle values are set to $\pi/2$ at corners and π at all other boundary vertices. While this algorithm is observed to behave well in practice, as mentioned earlier, further work is necessary to determine whether formal guarantees (regarding general existence and convergence) can be established.

In this context let us remark that our overall seamless parametrization construction does not in any way rely on the cone metric actually being conformal – this was merely a convenient natural choice that allows to easily ensure existence of a locally injective map with a given boundary curvature in the proof. If different approaches to conformal mapping, to other discrete notions of conformal mapping, or entirely different non-conformal methods for parametrization with piecewise straight boundaries are designed, these can be used alternatively.

A recently proposed conformal mapping method [Sawhney and Crane 2017] supporting cone and boundary curvature prescription is particularly efficient – but, for instance, does not include

remeshing capabilities inevitably required for full robustness. One could construct a hybrid solution with the more efficient algorithm tried first, and the robust but slower one serving as fallback.

Note that, if this is important in a use case, the edge flips which are performed by the conformal metric computation algorithm can ultimately be realized by means of edge splits, as described in [Fisher et al. 2007]. In this way the output mesh is a locally refined version of the input mesh (rather than a mesh with arbitrarily different combinatorial structure) and its embedding is preserved.

5.4.3 Equalization

We (arbitrarily) number the segments of $\partial M'$ and set up the system matrix A (5.6) accordingly, with one equation (5.3) for each pair of mates (or equation (5.5) where T-nodes are involved, cf. Sec. 5.3.2). The right-hand side $\boldsymbol{b}$ is determined by measuring the lengths of the segments under the metric computed in Sec. 5.4.2.

Then we solve the linear system $A\boldsymbol{w} = \boldsymbol{b}$. As it is underdetermined, we compute the least-norm solution $\boldsymbol{w}^*$ via $A^T A\boldsymbol{w} = A^T\boldsymbol{b}$. If the cut graph contains an extra path (cf. Sec. 5.3.2), we additionally fix the two padding width variables associated with the two extra path segments to zero (by eliminating them from A); Prop. 12 asserts that the system remains feasible.

The resulting solution $\boldsymbol{w}^*$ does not generally satisfy the important non-negativity constraint of problem (5.6). However, we can now add a sufficiently large constant (as described in the proof of Prop. 12) to all padding widths w_i^* (except those associated with an extra path, already fixed to zero). This ensures that the equalizing padding widths are non-negative.

5.4.4 Flattening

To obtain the map F from the conformal cone metric, we first need to extend the cut graph G to G^T (cf. Sec. 5.2.3). In each component of M' we pick a non-corner point p on a segment, and compute the shortest paths from p to all cones in the component. The union of these paths forms the tree T, extending G to G^T. Note that the piecewise-linear form of padding we use in the discrete setting (cf. Sec. 5.4.5) does not require the tree to meet the segment at right angles at root point p.

The conformal metric computed in Sec. 5.4.2 is flat on all of M^c, so its components can be laid out in the plane [Springborn et al. 2008], isometrically with respect to the metric, to obtain $F(M^c)$.

5.4.5 Padding

The padding operation described in Sec. 5.2.4 can be performed using piecewise-linear maps, as illustrated in Fig. 5.15. We start with inserting a straight (in flat/parametrization metric) line into the mesh M along a segment, requiring that no vertex is contained in the strip this line delineates. This line cuts the edges and creates new vertices; resulting polygons are triangulated. Then, stretch and lateral shift can be performed by moving those vertices that are on the segment. Note that, as no vertices lie in the strip, the mesh within the rectangular strip is a simple triangle strip and laterally translating the segment vertices does not cause triangle inversions as long as their order is preserved – which is the case with the reparametrization ϕ (cf. Appendix 5.10).

We note that the resulting seamless parametrization is not of immediate practical use: the scale distortion involved in the conformal map together with the additional padding-induced stretching often leads to high parametric distortion. This map, however, provides a valid (locally injective and seamless) starting point required by robust optimization methods that can convert it to a low-distortion parametrization (to the extend permitted by the cone prescription). We

emphasize that for non-convex problems, the ability to obtain a feasible starting point is critical: first, this is, in general, the only way to guarantee that a solution is found; second, this allows one to use robust optimization techniques that always stay in the feasible region during optimization.

5.4.6 Distortion Optimization

For the optimization of the seamless padded map, in our implementation, we use the symmetric Dirichlet energy together with efficient quadratic proxies as described in [Rabinovich et al. 2017b]. This method preserves local injectivity during optimization by design; we additionally include linear seamlessness constraints to preserve seamlessness of the map:

$$\vec{e}_i = R^{k_{ij}\frac{\pi}{2}}\vec{e}_j \quad \text{for each pair } (i, j) \text{ of identified mesh edges,}$$

where $\vec{e}_i$ is the edge vector of edge i in the parametric domain, and $R^{k_{ij}\frac{\pi}{2}}$ is a rotation by $k_{ij}\frac{\pi}{2}$, where the constant integer k_{ij} is determined by the edges' relative initial orientation in the domain.

We use the common symmetric Dirichlet objective in the optimization. In the context of specific applications, other application-dependent objectives, not focussing on distortion alone, may be relevant. For instance, for seamless global texturing, the seamless parametrization would have to be quantized (resulting in particular in an *integer grid map* with discrete translations in the transitions across cuts) [Ray et al. 2010]. The same is true for quad mesh generation or for constructing domains for spline spaces. Techniques to perform such quantization rely on using seamless parametrizations as a starting point [Campen et al. 2015; Lyon et al. 2019], of the type our method provides. It can also be useful to include directional terms in the objective, to support alignment to principal curvature or other directions. Finally, potential inaccuracies in the results' seamlessness due to numerical precision limits could be eliminated [Mandad and Campen 2019].

5.5 Examples

We demonstrate our implementation of the algorithm described in Sec. 5.1–5.6 on a number of examples.

Figure 5.17 shows a visualization of the seamless parametrizations constructed on models from the dataset provided by [Myles et al. 2014]. We employed the cone position and angle prescriptions included in this dataset. Figure 5.16 demonstrates the algorithm handling topologically complex surfaces as well as randomly prescribed singularities. In all cases locally injective seamless global parametrizations were obtained. Note that seamless does not mean that cuts are not visible in these checkerboard visualizations; for this the maps would additionally need to be quantized [Campen et al. 2015] – a process for which our method provides a suitable initialization.

To explore the numeric limits of our implementation, we applied it to N-tori, for increasing N. For an 80-torus, as depicted in Figure 5.18, the implementation succeeds; for a 100-torus we are still able to obtain an initial seamless map – however, with a level of distortion that state-of-the-art local injectivity preserving optimization methods prove to have trouble with, due to numerical precision issues. For even larger N, the computation of the constrained conformal map starts to suffer from occasional numerical issues (e.g. step size going down to numerical zero) as well. Investigation of the numerical aspects of map optimization in high distortion cases is an important direction for future research.

5.6 Conclusion and Future Work

This paper provides a general path to obtaining seamless parametrizations with a given set of cones. On a conceptual level, the approach is simple: pad a parametrization that maps cut segments to straight lines, with padding determined by solving a linear system. Our algorithm demonstrates that for (almost) any user-specified or automatically computed choice of cones a

corresponding global parametrization can be constructed, without introducing additional cones.

A limitation of our approach in its current form is that it does not take into account the holonomy angles on global homology loops (in addition to cone angles), which, for instance, is important for parametrizations following a global guiding field. We expect that by using different forms of cut graph construction, based on given global holonomy angles, many of the ideas will be applicable to such a setting as well; we plan to address this in a separate paper.

Other directions of future work include generalization to surfaces with boundaries as well as aligning to tagged feature curves or other prescribed directions on the surface.

In the smooth setting, we have constructively shown the existence of a locally injective, seamless parametrization. In practice, numerical optimization routines bring about additional challenges related to precision limits, which here affects the discrete conformal map computation. A potential path could be the replacement of this initial map computation with a different technique – perhaps exploiting the fact that conformality is not actually required.

5.7 Illustrative Example

We consider the simplest example: a torus with two cones, $k_0 = 2$, $k_1 = 6$, i.e., cone angles π and 3π, shown in Figure 5.19.

Cut Graph We cut the surface into a 2-corner and a 6-corner component (cf. Fig. 5.19 top). The cut graph was embedded in the surface in such a way that the total cone curvature contained in each component is compatible with the number of corners in terms of Gauss-Bonnet: a cone with $k_0 = 2$ lies in the 2-corner region, a cone with $k_1 = 6$ in the 6-corner region.

Cone Metric We compute a cone metric on each of the two components (e.g. conformal, given by a pointwise scale factor) which is flat everywhere except at the cones, where it has the prescribed curvature. In addition, we require the boundary to be geodesically straight at all boundary

points except for the corners, where it forms right angles under the metric.

Metric to Parametrization If we add cuts connecting all cones to the boundary (indicated with dashed curves) this cone metric is flat in the interior and corresponds to a global parametrization of the torus, with two charts (cf. Fig. 5.19 middle left and right). The image of each of the two maps is a domain with rectilinear boundary, consisting of straight segments meeting at right angles (excluding the cuts to the cones). As the angle between any two segments is an integer multiple of $\pi/2$, this parametrization is *rotationally* seamless, but it may have a jump in scale across cuts. In particular, two segments corresponding to the same cut graph branch – here (1,7), (2,5), (3,6), and (4,8) – may have different lengths in general.

Equalization by Padding To obtain a seamless parametrization we equalize the lengths of identified pairs of segments. This is achieved by adding *padding*, i.e., we extend the parametric domain by shifting straight segments in orthogonal direction (cf. Fig. 5.19 bottom). For each segment i, numbered sequentially around each component, ℓ_i, $i = 1 \dots 8$, is its parametric length. For a segment i, after padding its length becomes $\ell_i + w_{\text{prev}(i)} + w_{\text{next}(i)}$, where $\text{prev}(i)$ and $\text{next}(i)$ are previous and next segment indices around the component, and w_j is the padding width for segment j. Equating the post-padding lengths of all four pairs of identified segments yields the following four equations in this example:

$$\ell_1 + w_2 + w_6 = \ell_7 + 2w_8; \quad \ell_2 + w_1 + w_3 = \ell_5 + w_4 + w_6;$$

$$\ell_3 + w_2 + w_4 = \ell_6 + w_1 + w_5; \quad \ell_4 + w_3 + w_5 = \ell_8 + 2w_7$$

where ℓ_i are the known segment lengths, w_i are the unknown padding widths. The matrix of this equation system has the form

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & -2 \\ 1 & 0 & 1 & -1 & 0 & -1 & 0 & 0 \\ -1 & 1 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & -2 & 0 \end{bmatrix}$$

and the right-hand side is $\boldsymbol{b} = [\ell_7 - \ell_1,\ \ell_5 - \ell_2,\ \ell_6 - \ell_3,\ \ell_8 - \ell_4]^T$, i.e., parametric length mismatches of identified segments. To *equalize* segments lengths, we need to find a solution of the system $A\boldsymbol{w} = \boldsymbol{b}$, where $\boldsymbol{w}$ is the vector of padding widths, such that $\boldsymbol{w} \geq 0$. This non-negativity condition is important to guarantee that the domain does not degenerate through padding. Observe that A has full (row) rank, which ensures that the system has a (possibly non-unique) solution. Observe further that $A\mathbf{1} = 0$ in this case, i.e., after computing an arbitrary solution, we can obtain a non-negative solution by adding a sufficiently large constant. More generally, note that A is determined solely by the choice of the cut graph combinatorics. For instance, without the blue or without the yellow branch, the cut graph (cutting the surface to a single topological disk in these cases) would yield a system that does *not* have a non-negative solution $\boldsymbol{w}$ for every possible $\boldsymbol{b}$.

Seamless Parametrization Once the padded domain is obtained, we remap the original image onto this domain. This is done by stretching outwards thin strips running along the segments to cover the added space in the rectangular regions padded onto the domain, yielding a seamless global parametrization.

5.8 Proofs of Equalizability

5.8.1 Genus 3+

This proof is constructive, yet it is only intended to prove feasibility. In practice, a simple linear system solve can be used to obtain a solution instead (cf. Sec. 5.4.3), while our theorem ensures that this linear system solve always succeeds.

Non-Fourfold Case

The hole chain cut graph G, together with the extra path, cuts M into two components M'_0 and M'_1, neither of which has its number of segments m_k divisible by 4. The boundary of at least one of these components contains a segment of a hole, located between the two parts of an odd-couple (cf. Sec. 5.3.1); let this component be M'_1.

Our proof is based on the observation (Lemma 5.10) that for each of the domains M'_k, it is possible to attain *arbitrary* target segment lengths $\tilde{\ell}_i$, $i = 0 \dots m_k - 1$, using (possibly negative) padding. Hence it is, in particular, possible to choose padding widths such that lengths of mated segments match (Lemma 5.12).

If there were no T-nodes, such (possibly negative) equalizing padding widths could easily be transformed into non-negative ones: adding a sufficiently large constant c to each yields non-negative padding widths while preserving equalization (each segment length increases by $2c$). Cut graphs with T-nodes result in complex segments on the cut, and these are not mated in pairs. Equalization is preserved if we add c to all padding widths *except* those of the two extra path segments – but then these two would remain possibly negative. This requires performing two intermediate modifications:

First, we make both extra path segments' padding widths non-negative (Lemma 5.13). This leads to one equalization equation being violated. We then show that, exploiting the presence of

an odd-couple, a further modification of padding widths (Lemma 5.14) can restore equalization (Prop. 12).

REMARK: While it may be possible to construct cut graphs without T-nodes for any configuration of genus and cones, and thereby simplify the proof, this would require the consideration of (possibly many) further special cases, with more than two components, depending on genus and cone curvatures. We choose the version leading to a simpler construction algorithm with fewer special cases.

We first establish two auxiliary results for an individual component M'_k, omitting the subscript k. We number segments cyclically along the component's boundary, with numbers from 0 to $m-1$. Let $B\boldsymbol{w} = \boldsymbol{d}$, with $m \times m$ matrix B, be the *length adjustment* system formed by equations $w_{i-1} + \ell_i + w_{i+1} = \tilde{\ell}_i$ (with index arithmetic done $\mod m$), for initial lengths ℓ_i and arbitrary target lengths $\tilde{\ell}_i$.

Lemma 5.10. *There are padding widths $w_i \in \mathbb{R}$ satisfying $B\boldsymbol{w} = \boldsymbol{d}$.*

Proof. The system matrix B is an $m \times m$ *circulant* matrix, with *associated polynomial* $f(x) = x + x^{m_k-1} = x(x^{m_k-2}+1)$ [Davis 2012]. It is full rank whenever $f(e^{j/m_k 2\pi\mathrm{i}}) \neq 0$, for $j = 0 \ldots m_k-1$, as its determinant is given by the product of these values (f on m^{th} roots of unity). It is straightforward to check that $f(e^{j/m 2\pi\mathrm{i}}) = 0$ (for some j) requires $e^{j(m-2)/m_k 2\pi\mathrm{i}} = -1$. This is equivalent to $2j(m-2)/m$ being odd, which in turn, requires $4j/m$ to be odd, i.e., m must be a multiple of 4, contradicting the assumption $m \bmod 4 \neq 0$. □

Lemma 5.11. *For an arbitrary choice of index $0 \leq j < m$ and an odd number p, there are padding widths $w_i \in \mathbb{R}$ with $w_j = 1$ such that all equations of $B\boldsymbol{w} = \boldsymbol{0}$, except the $(j+p \mod m)$-th equation, are satisfied.*

Proof. Choose $w_{j+p+1}, w_{j+p+3}, w_{j+p+5}, \ldots, w_{j+p-1}$ alternatingly as 1 and -1 (where index summation

is done $\bmod m$). Note that if m is even, this sequence contains every other padding width, if m is odd, it contains every padding width, as illustrated below. In both cases, due to $m \bmod 4 \neq 0$, this sequence is of odd length. Thus, $w_{j+p+1} = w_{j+p-1}$. As p is odd, w_j is part of the sequence; we choose the alternating sign such that $w_j = 1$. As each segment $i \neq j+p$ has its previous and next segments either both not padded or padded with alternating signs, its length is not changed, i.e., padded length $\ell'_i = \ell_i$, but $\ell'_{j+p} = \ell_{j+p} \pm 2$. □

Combining the result of Lemma 5.10 for both components yields the following lemma.

Lemma 5.12. *The equalization system $A\boldsymbol{w} = \boldsymbol{b}$ induced by a hole chain cut graph with admissible extra path has a (possibly negative) solution $\boldsymbol{w}$.*

Proof. Choose the target length $\tilde{\ell}_i = 1$ for each simple segment i (consisting of one boundary curve), and $\tilde{\ell}_i = r$ for each complex segment consisting of r boundary curves (with $r-1$ flat joints). Note that for our hole chain cut graph $r = 2$ or $r = 3$ (when $m_k = 2$ for some k). According to Lemma 5.10, there are padding widths $\boldsymbol{w}_0$ for M'_0 and $\boldsymbol{w}_1$ for M'_1 such that padded lengths $\ell'_i = \tilde{\ell}_i$, and because lengths $\tilde{\ell}_i$ match for mated segments, these padding widths $\boldsymbol{w} = [\boldsymbol{w}_0, \boldsymbol{w}_1]$ are equalizing, thus $A\boldsymbol{w} = \boldsymbol{b}$. □

We now show that this initial, possibly negative solution can, in multiple steps, be transformed into a non-negative solution.

Lemma 5.13. *Consider the equalization system $A\boldsymbol{w} = \boldsymbol{b}$ induced by a hole chain cut graph with admissible extra path. Let a, b be the segments of the extra path, q an arbitrary hole segment of M'_1. There are padding widths $\bar{\boldsymbol{w}}$ with $\bar{w}_a = \bar{w}_b = 0$ such that all equalization equations of $A\bar{\boldsymbol{w}} = \boldsymbol{b}$ except for the one containing ℓ_q are satisfied.*

Proof. Lemma 5.12 yields padding widths satisfying $A\boldsymbol{w} = \boldsymbol{b}$, but possibly with $w_a \neq 0$ or $w_b \neq 0$. Let $w'_i = w_i$, for all i, except for $w'_a = 0$, $w'_b = w_b + w_a$ (i.e. we move all extra path padding to one

side of the path). The equation $A\boldsymbol{w}' = \boldsymbol{b}$ still holds, because the padding widths of segments a and b do not appear individually, but only as sum in these equations (5.5), and $w'_a + w'_b = w_a + w_b$.

Observe that due to b being an extra path and q a hole segment index (and the extra path being connected to a hole, cf. Def. 5.8), at least one of the two (cw or ccw) cyclic index distances along the boundary of M'_1 is odd, thus there is an odd number p such that $(b+p \mod m_1) = q$. We now apply Lemma 5.11 to the component M'_1 to make the padding width of b zero. For the choice $j = b$ and the odd number p, this lemma yields padding widths w''_i for M'_1 (for each segment i of M'_0, we set $w''_i = 0$) with $w''_b = 1$ which leave all segment lengths unchanged except for that of segment q. We obtain padding widths satisfying the lemma as $\bar{w}_i = w'_i - w'_b w''_i$. □

Now we show that a further modification allows us to find padding width that also satisfies the equation containing ℓ_q. This is the only step that requires using the assumption that the hole chain cut graph contains an odd-couple (cf. Sec. 5.3.1).

Lemma 5.14. *Suppose the equalization system $A\boldsymbol{w} = \boldsymbol{b}$ is induced by a hole chain cut graph with admissible extra path and an odd-couple. Let q be an arbitrary hole segment of M'_1 between the mated segments d and e of the odd-couple in the hole chain. Then for any $\delta \in \mathbb{R}$ there are padding widths $\boldsymbol{w}$ with $w_a = w_b = 0$ such that the padded length $\ell'_q = \ell_q + \delta$, while all other lengths remain unchanged or are changed by the same amount for each group of mated segments, i.e., $A\boldsymbol{w} = \boldsymbol{0}$ except for the equation including ℓ_q.*

Proof. Let $d_0, \ldots, d_r$ be the sequence of connectors of the hole chain between an odd-couple pair of holes h_d, h_e. Note that r is odd. Each connector has two sides; let $c_0, \ldots, c_r$ be the sides on that side of the hole chain where the hole segment q is located. Then there are c_i, c_{i+1} such that segment q lies between them (Fig. 5.20).

Let $v_{c_j} = -1^{i-j}\frac{1}{2}\delta$, $j = 0, \ldots, i$, and $v_{c_j} = -1^{i-j+1}\frac{1}{2}\delta$, $j = i+1, \ldots, r$. Each connector side c_j corresponds to one segment s_{c_j} or, in case the extra path connects to such a side and splits it, two

segments s_{c_j} and $\bar{s}_{c_j}$ (for example, c_4 in the figure below). Let $w_{s_j} = v_{c_j}$ (and in case of a connector split by the extra path also $w_{\bar{s}_j} = v_{c_j}$) for each c_j, and zero for all other segments not involved in connector sides $c_0, ..., c_r$ (thus in particular the extra path segments a, b). With these padding widths, we have $\ell'_q = \ell_q + \delta$, the lengths of mated segments d and e are either both increased or both decreased by the same amount $\frac{1}{2}\delta$. All other hole segments between d and e have their previous and next connector segment padded with opposite signs, preserving their lengths $\ell'_i = \ell_i$ – unless the extra path is connected to one of these holes: suppose it is connected to the hole between c_j and c_{j+1}, then there are two hole segments (separated by the extra path) between c_j and c_{j+1}, and one's length is increased by $\frac{1}{2}\delta$, the other's decreased by $\frac{1}{2}\delta$; as they (due to the T-node at the extra path) are mated in combination, these values cancel in the equalization equations. □

Finally, we obtain a non-negative solution of the equalization system, as required.

Proposition 12. *The equalization system $A\boldsymbol{w} = \boldsymbol{b}$ induced by a hole chain cut graph with admissible extra path and an odd-couple, has a non-negative solution $\boldsymbol{w}$.*

Proof. Let q be a hole segment of M'_1 that lies between two segments of an odd-couple. For this q, Lemma 5.13 asserts there are padding widths $\bar{\boldsymbol{w}}$ with $\bar{w}_a = \bar{w}_b = 0$, satisfying all but one equalization equation. Let δ be the error, i.e., the difference between the padded length ℓ'_q and the padded length of q's mate(s). For these q and δ, Lemma 5.14 yields padding widths $\hat{w}_i$ such that $w'_i = \bar{w}_i + \hat{w}_i$ satisfy all equations $A\boldsymbol{w}' = \boldsymbol{b}$, because with this addition, the padded length ℓ'_q of segment q is adjusted to cancel the error δ, while equalization of all other mated segments is preserved, and $w'_a = w'_b = 0$. Now non-negative equalizing padding widths can be derived: Let $\mathbf{1}'$ be the vector of ones, except for two zeroes at entries a and b. Then $A\mathbf{1}' = \mathbf{0}$, because each equation (5.3)/(5.5), thus each row of A, contains *two* (not necessarily distinct) padding width variables with positive sign and *two* padding width variables with negative sign – and possibly further entries $-w_a$, $-w_b$, which, however, are zero in $\mathbf{1}'$. Let $\lambda = \min w'_i$. Then padding widths $\boldsymbol{w} = \boldsymbol{w}' - \lambda\mathbf{1}'$ are non-negative and, due to $A\boldsymbol{w} = A\boldsymbol{w}' - \lambda A\mathbf{1}' = \boldsymbol{b}$, equalizing. □

Fourfold Case

In this case, the cut graph contains no extra cut path and we have a single component M' with the number of segments m divisible by 4. As shown in the proof of Lemma 5.10, system matrix B does not have full rank in this case. Its upper left $(m-2)\times(m-2)$-submatrix, however, has full rank (as it is a tridiagonal Toeplitz matrix), thus B^-, which is B with the last two rows removed, is a (rectangular) matrix with full row rank. This implies we can obtain padding widths $\boldsymbol{w}$ with $B^-\boldsymbol{w} = \mathbf{1} - \boldsymbol{\ell}^-$, i.e., they bring all segments *but the last two* to unit length.

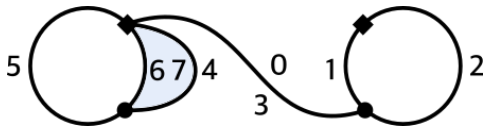
In the case of fourfold cones, the transitions across the cut graph extension T (cf. Sec. 5.2.3) are rotations by a multiple of 2π (= identity); as a consequence, the cut extension T can actually be omitted. This implies that the boundary of the flattening $F(M')$ is formed exclusively by the segments and is entirely rectilinear, as illustrated in Fig. 5.21 right. Without the loss of generality, we assume that all even-index segments are laid out horizontally and all odd-index segments vertically. Counterclockwise around the boundary $\partial F(M')$, horizontal segments alternate between positive and negative u-directions, and vertical segments alternate between positive and negative v-direction. The fact that $\partial F(M')$ is (and after padding remains) a *closed* polygon then implies

$$\sum_{i=0,\dots,m/2-1} -1^i \ell_{2i} = 0, \qquad \sum_{i=0,\dots,m/2-1} -1^i \ell_{2i+1} = 0.$$

This, in turn, implies that if all even/odd segments but one have unit length, the remaining one has unit length as well. Hence, the last two conditions of B are, in the fourfold case, satisfied automatically if all other conditions are satisfied. We conclude that $B^-\boldsymbol{w} = \mathbf{1}-\boldsymbol{\ell}^-$ implies $B\boldsymbol{w} = \mathbf{1}-\boldsymbol{\ell}$. The vector $\boldsymbol{w}$ may contain negative values, but we can add an arbitrary constant shift $\boldsymbol{w}^* = \boldsymbol{w}+\lambda\mathbf{1}$ because $B\boldsymbol{w}^* = \mu\mathbf{1} - \boldsymbol{\ell}$ (with $\mu = (2\lambda+1)$), leading to $A\boldsymbol{w}^* = \boldsymbol{b}$ and $\boldsymbol{w}^* \geq 0$ for a sufficiently large λ. □

5.8.2 Genus 1

The equalization system for the genus 1 cut graph pattern is:



$$
\begin{aligned}
w_1 + w_5 - w_2 - w_4 &= \ell_3 - \ell_0 \\
w_6 + w_6 - w_3 - w_5 &= \ell_4 - \ell_7 \\
w_0 + w_2 - w_7 - w_7 &= \ell_6 - \ell_1 \\
w_1 + w_3 - w_0 - w_4 &= \ell_5 - \ell_2
\end{aligned}
$$

One can easily verify that the system matrix has full row rank, and that it has positive vectors (e.g. $\mathbf{1}$) in its kernel, thus has a non-negative solution for any righthand side.

5.8.3 Genus 2

The five different cut graph patterns covering all possible cone choices for genus 2 are depicted in Fig. 5.22. One can easily verify explicitly that their equalization system matrices all have full row rank, and that they have positive vectors (e.g. $\mathbf{1}$) in their kernel, thus have non-negative solutions for any righthand side.

2 corners:

$$
\begin{aligned}
w_1 + w_7 - w_{13} - w_{19} &= \ell_{12} - \ell_0 \\
w_1 + w_3 - w_{17} - w_{19} &= \ell_{18} - \ell_2 \\
w_5 + w_7 - w_{13} - w_{15} &= \ell_{14} - \ell_6 \\
w_8 + w_{10} - w_{20} - w_{22} &= \ell_{23} - \ell_{11} \\
w_0 + w_2 - w_0 - w_6 &= \ell_7 - \ell_1 \\
w_2 + w_{17} - w_{11} - w_{16} &= \ell_8 - \ell_3 \\
w_{20} + w_{22} - w_{21} - w_{21} &= \ell_9 - \ell_4 \\
w_6 + w_{15} - w_{11} - w_{16} &= \ell_{10} - \ell_5 \\
w_{12} + w_{14} - w_{12} - w_{18} &= \ell_{19} - \ell_{13} \\
w_5 + w_{14} - w_4 - w_{23} &= \ell_{20} - \ell_{15} \\
w_8 + w_{10} - w_9 - w_9 &= \ell_{21} - \ell_{16} \\
w_3 + w_{18} - w_4 - w_{23} &= \ell_{22} - \ell_{17}
\end{aligned}
$$

7 corners:

$$
\begin{aligned}
w_1 + w_{23} - w_{11} - w_{13} &= \ell_{12} - \ell_0 \\
w_2 + w_4 - w_8 - w_{10} &= \ell_9 - \ell_3 \\
w_{14} + w_{16} - w_{20} - w_{22} &= \ell_{21} - \ell_{15} \\
w_0 + w_7 - w_0 - w_6 &= \ell_{23} - \ell_1 \\
w_3 + w_{17} - w_{18} - w_{21} &= \ell_{22} - \ell_2 \\
w_3 + w_{17} - w_{16} - w_{21} &= \ell_{20} - \ell_4 \\
w_7 + w_{11} - w_8 - w_{10} &= \ell_{19} - \ell_5 \\
w_{13} + w_{23} - w_{14} - w_{22} &= \ell_{18} - \ell_6 \\
w_1 + w_5 - w_2 - w_4 &= \ell_{17} - \ell_7 \\
w_9 + w_{19} - w_{15} - w_{20} &= \ell_{16} - \ell_8 \\
w_9 + w_{19} - w_{15} - w_{18} &= \ell_{14} - \ell_{10} \\
w_5 + w_{12} - w_6 - w_{12} &= \ell_{13} - \ell_{11}
\end{aligned}
$$

3 corners:

$$
\begin{aligned}
w_1 + w_{19} - w_9 - w_{11} &= \ell_{10} - \ell_0 \\
w_5 + w_7 - w_{16} - w_{18} &= \ell_{17} - \ell_6 \\
w_2 + w_4 - w_{13} - w_{15} &= \ell_{14} - \ell_3 \\
w_0 + w_9 - w_0 - w_8 &= \ell_{19} - \ell_1 \\
w_3 + w_{11} - w_{12} - w_{17} &= \ell_{18} - \ell_2 \\
w_3 + w_{13} - w_{12} - w_{17} &= \ell_{16} - \ell_4 \\
w_6 + w_7 - w_8 - w_{14} &= \ell_{15} - \ell_5 \\
w_5 + w_6 - w_4 - w_{14} &= \ell_{13} - \ell_7 \\
w_{15} + w_{19} - w_{16} - w_{18} &= \ell_{12} - \ell_8 \\
w_1 + w_{10} - w_2 - w_{10} &= \ell_{11} - \ell_9
\end{aligned}
$$

5 corners:

$$
\begin{aligned}
w_1 + w_{15} - w_7 - w_9 &= \ell_8 - \ell_0 \\
w_2 + w_4 - w_4 - w_6 &= \ell_5 - \ell_3 \\
w_{10} + w_{12} - w_{12} - w_{14} &= \ell_{13} - \ell_{11} \\
w_0 + w_{10} - w_0 - w_9 &= \ell_{15} - \ell_1 \\
w_3 + w_6 - w_7 - w_{13} &= \ell_{14} - \ell_2 \\
w_3 + w_5 - w_{11} - w_{13} &= \ell_{12} - \ell_4 \\
w_2 + w_5 - w_1 - w_{11} &= \ell_{10} - \ell_6 \\
w_8 + w_{14} - w_8 - w_{15} &= \ell_9 - \ell_7
\end{aligned}
$$

6 corners:

$$
\begin{aligned}
w_1 + w_{10} - w_{17} - w_{26} &= \ell_{16} - \ell_0 \\
w_2 + w_4 - w_{23} - w_{25} &= \ell_{24} - \ell_3 \\
w_7 + w_9 - w_{18} - w_{20} &= \ell_{19} - \ell_8 \\
w_{11} + w_{14} - w_{27} - w_{30} &= \ell_{31} - \ell_{15} \\
w_0 + w_{22} - w_0 - w_{21} &= \ell_{10} - \ell_1 \\
w_3 + w_{29} - w_8 - w_{28} &= \ell_9 - \ell_2 \\
w_3 + w_{29} - w_{15} - w_{30} &= \ell_{11} - \ell_4 \\
w_{22} + w_{26} - w_{23} - w_{25} &= \ell_{12} - \ell_5 \\
w_{17} + w_{21} - w_{18} - w_{20} &= \ell_{13} - \ell_6 \\
w_8 + w_{28} - w_{15} - w_{27} &= \ell_{14} - \ell_7 \\
w_6 + w_{16} - w_5 - w_{16} &= \ell_{26} - \ell_{17} \\
w_{13} + w_{19} - w_{12} - w_{24} &= \ell_{25} - \ell_{18} \\
w_{13} + w_{19} - w_{14} - w_{31} &= \ell_{27} - \ell_{20} \\
w_6 + w_{10} - w_7 - w_9 &= \ell_{28} - \ell_{21} \\
w_1 + w_5 - w_2 - w_4 &= \ell_{29} - \ell_{22} \\
w_{12} + w_{24} - w_{11} - w_{31} &= \ell_{30} - \ell_{23}
\end{aligned}
$$

5.9 Proof of Cone Metric Existence

[Troyanov 1991] presents a general proof of cone metric existence on closed surfaces. It "extends [...] to surfaces with (piecewise geodesic) boundary", but the extension is not spelled out. [Cherrier 1984] focuses on the case with boundary, but does not specifically consider the relevant delta distributions of curvature. We provide a proof tailored to our setting.

Let M' be one of the disk-topology connected components of the cut surface (we will drop

the index of the component in the following). Consider the expansion M'_{exp} of M', obtained by joining a copy of the geodesic disk of size ϵ_p in M centered at $\pi(p)$, to each boundary point of M'. Multiple $p \in \partial M'$ corresponding to the same $\pi(p)$ get separate copies of the disk centered at $\pi(p)$ and ϵ_p is chosen sufficiently small for each p so that M'_{exp} still has disk topology.

To simplify the exposition, we assume that on the surface M the branches of the cut form right angles – the proof can be extended to arbitrary angles, as long as the curves are transversal, but requires a more complex solution ϕ_1 below, with additional cones at the corners, as explained in more detail in [Bunin 2008].

Consider a conformal map f from M'_{exp} to the plane (e.g. to a disk). As M'_{exp} has disk-topology, such a map exists. As M' is in the interior of M'_{exp} the map is conformal at the points of the boundary $\partial M'$. The conformal scale factor $|f'|$, where f' is the complex derivative of the map expressed in local complex coordinates on the tangent plane, defines the conformal metric on $\text{Int}(f(M'_{\text{exp}}))$, in particular, on all of $f(M') = M''$ including the boundary. Let γ_i, $i = 1 \dots m$, be the curves of the boundary of M''; these curves are smooth, as the boundary of M' is smooth, and meet at right angles.

We now construct on M'' a metric with the desired properties; then the metric on M' is obtained by a pullback through f. As M'' is flat, the equation for the metric in the interior points x of M simplifies to

$$\Delta\phi = \sum_j \hat{\Theta}_j \delta(f(p_j) - x),$$

where $\hat{\Theta}_j$ is the target curvature at cone $c_j = (p_j, \hat{\Theta}_j)$. If the geodesic curvature at non-corner boundary points is given by a smooth function κ, then we have the Neumann boundary condition

$$\frac{\partial\phi}{\partial n} = -\kappa,$$

that needs to be satisfied to obtain straight boundary edges in the final metric. Note that κ may be

discontinuous at the corner points but it is still in L_2. We can find a particular solution u_1 satisfying the Poisson equation on M'' without boundary conditions directly as $\phi_1 = \sum_j \hat{\Theta}_j \ln(|z - f(p_j)|)$ with singularities at $f(p_j)$.

Then we solve the Laplace equation $\Delta\phi_2 = 0$, for ϕ_2 with smooth Neumann conditions $\partial\phi_2/\partial n = \kappa - \partial\phi_1/\partial n$. For this problem to have a solution, the Neumann boundary condition needs to integrate to zero over the boundary. Observe that because the domain M'' is flat, the integral of the geodesic curvature κ over the boundary, with the sum of corner angles $n\frac{\pi}{2}$ added, must be 2π, i.e., $\int_{\partial\Omega} \kappa ds = 2\pi - n\frac{\pi}{2}$. In addition, $\int_{\partial\Omega} \partial\phi_2/\partial n ds = \sum_j \hat{\Theta}_j$, by the Gauss theorem. Finally, note that by the cut graph admissibility assumption on the number of corners, $2\pi - n\frac{\pi}{2} - \sum_j \hat{\Theta}_j = 0$, i.e., the integral condition for the Neumann problem is satisfied. Therefore, the problem has a unique, up to a constant, solution. This solution is in H^2 (and, by Sobolev Lemma, C^0 up to the boundary) for domains with piecewise smooth boundary and convex corners between curves (cf. [Grisvard 1985, p. 174]). The sum $\phi = \phi_1 + \phi_2$ satisfies the Poisson equation and boundary conditions. The metric ϕ is nonsingular at the boundary, therefore it is conformal, and the angles between boundary curves are preserved. We conclude that the pullback of this metric to M' is the needed metric.

5.10 Map Padding

As laid out in Sec. 5.2.4, map padding consists of the application of stretch maps to rectangular regions, and lateral shifts within these. To define these precisely, we, w.l.o.g., consider the case of a horizontal segment s_j (aligned with the u-axis in (u, v) coordinates) to be padded by w_j in positive v direction, as illustrated in Fig. 5.6 – the other cases (negative v, and positive/negative u) are handled analogously.

In the case of a segment split by T, we assume that T (which can be chosen freely) meets the segment at a right angle with a straight cut in the parametric domain. Then both parts can be

treated separately using the following operations without special case handling – except for the same rectangle thickness being used for both parts.

STRETCH Let τ_j be the thickness (here: the height) of rectangle R_j, and $(u_{j\min}, v_{j\min})$ the coordinates of the lower left corner of R_j. The map g_i applied to the strip to perform the stretching is a simple one-dimensional scaling by factor $o_j = \frac{w_j+\tau_j}{\tau_j}$:

$$g_j : (u, v) \mapsto (u, v_{j\min} + o_j(v - v_{j\min})). \tag{5.7}$$

SHIFT We apply a deformation (lateral shift) within a rectangle R_j^p that leads to a (piecewise) constant speed parametrization of the segment s_j. We use a simple blend (linear in v) between the map $\phi_j : [u_{j\min}, u_{j\max}] \to [u_{j\min}, u_{j\max}]$ that reparametrizes segment s_j to (piecewise) constant speed (applied at the top of the strip) and the identity map $u \mapsto u$ (applied at the bottom):

$$r_j : (u, v) \mapsto (t\phi_j(u) + (1 - t)u, v), \tag{5.8}$$

where $t = (v - v_{j\min})/\tau_j$ is the normalized relative v-coordinate within R_j^p. ϕ_j is a constant speed reparametrizaton for simple segments. For complex segments it is with piecewise constant speed, constant per boundary curve the segment consists of, such that the lengths of these boundary curves after reparametrization are in the same ratio as the padded lengths of their mates.

It is easy to verify that r_j is injective: the determinant of its Jacobian is

$$\det J(u, v) = (\partial\phi_j/\partial u(u) - 1)\, t + 1,$$

and due to $0 \le t \le 1$ and $\partial\phi_j/\partial u(u) > 0$ (as the scaled arc-length reparametrization is non-degenerate and orientation preserving) it is always positive.

Proof of Proposition 6. F is rotationally seamless, in particular locally injective and continuous

(on M^c). If $F^{p,m}$ is continuous, so is $f^{(p,m+1)}$ because g_{m+1} is continuous and it is identity on the interface between S_{m+1} and the rest of M^c. If $F^{s,m}$ is continuous, so is $F^{(s,m+1)}$ because r_{m+1} is continuous and it is identity on the interface between S_{m+1} and the rest of M^c. It follows that F^s is continuous. Analogously, as g_j and r_j are injective, local injectivity is preserved for F^s. Both types of maps, g_j and r_j, preserve the straightness and the orientation of all segments and therefore the pairwise angles between them, thus F^s is rotationally seamless like F. As angles between boundary curve images are not affected, cone angles are preserved as well. Each boundary curve segment which s_j consists of is parametrized with constant speed by $F^{s,j}$ by construction. As s_k with $k \neq j$ is identity on s_j (more precisely: that part of s_j contained in R_k and thus potentially affected by s_k), $F^s(s_j) = F^{s,j}(s_j)$. □

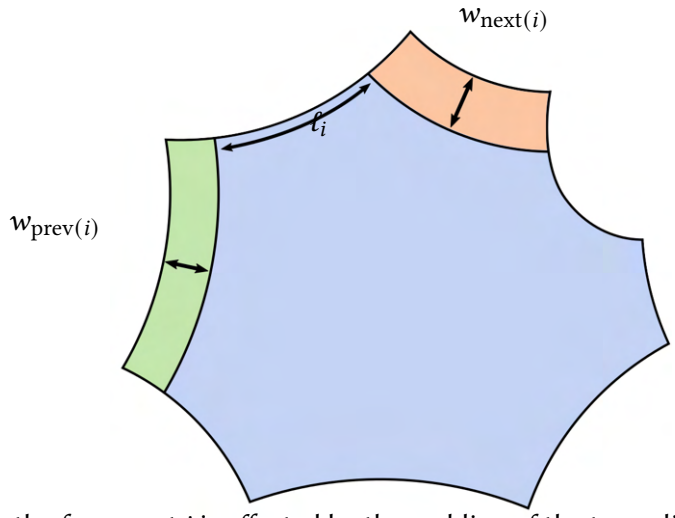


Figure 5.9: The length of segment i is affected by the padding of the two adjacent segments: the original length ℓ_i changes to $\ell_i + w_{\text{prev}(i)} + w_{\text{next}(i)}$.

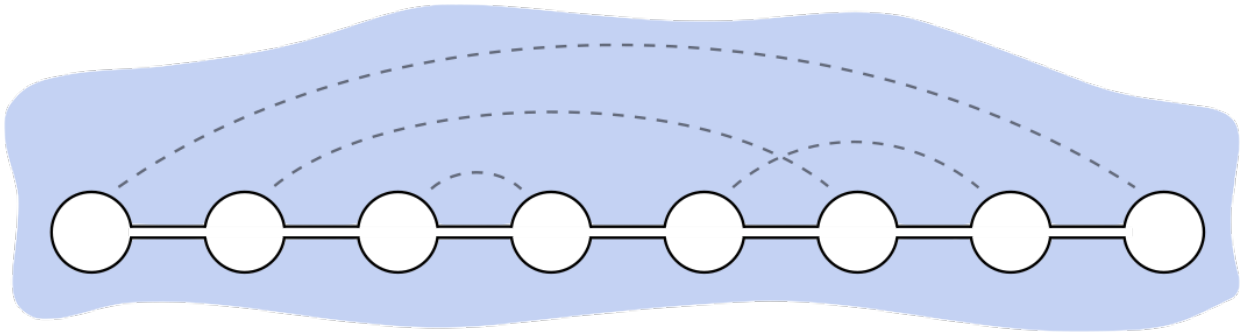


Figure 5.10: Schematic depiction of a chain of holes for a genus $g = 4$ surface: circles are holes (obtained by cutting the surface along g loops), straight line segments are the sides of cut paths (connectors) between these holes. Together, the hole chain cuts the surface to a topological disk (blue), i.e., a sphere with one hole (white, bounded by the black curve). An example of a hole partner correspondence is indicated by dashed arcs; depending on the chosen ordering of holes in the chain, these partner arcs will look different.

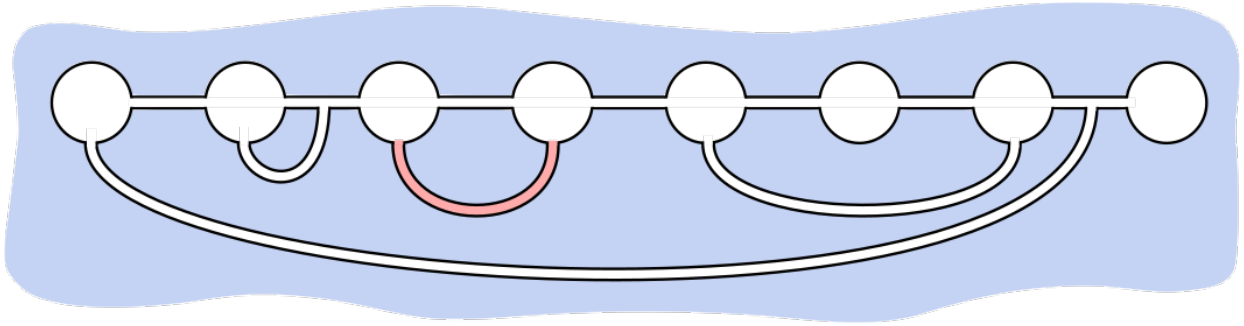


Figure 5.11: Examples of extra paths (bottom) that could be added to the hole chain cut graph. The red path is not an admissible extra path because it splits the surface into two components with $m_0 = 4$ and $m_1 = 8g - m_0 = 28$ corners (cf. Sec. 5.3.2).

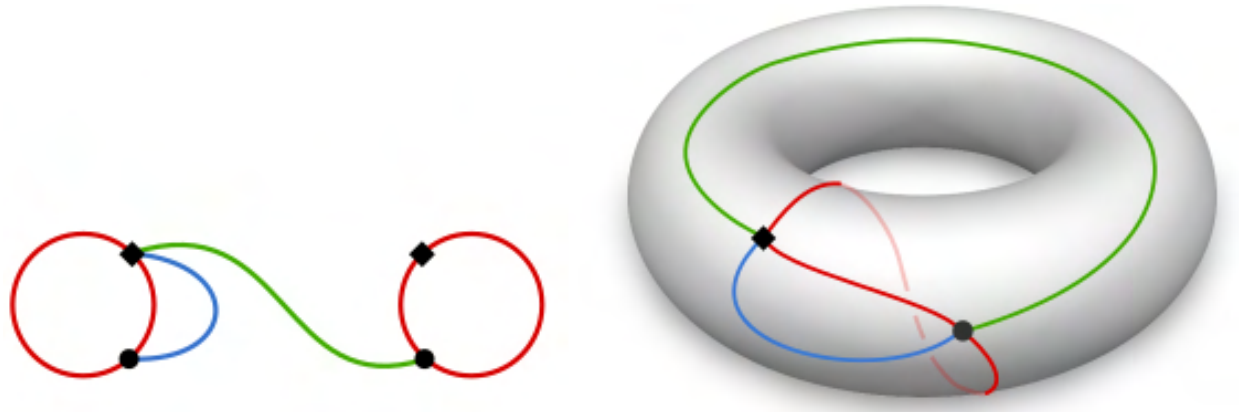


Figure 5.12: Cut Graph pattern for genus 1 surfaces, shown abstracly (left) and on an example surface (right). The surface is partitioned into a 2-corner region (enclosed by blue and red paths) and a 6-corner region.

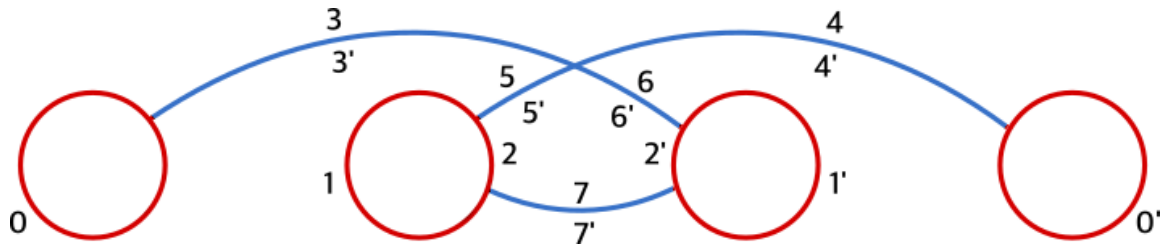


Figure 5.13: One of the cut graph patterns for genus 2 surfaces. Segments i and i' are mates, i.e., correspond to a common cut graph branch. The surface is partitioned into a 5-corner region (center) and a 11-corner region (surround).

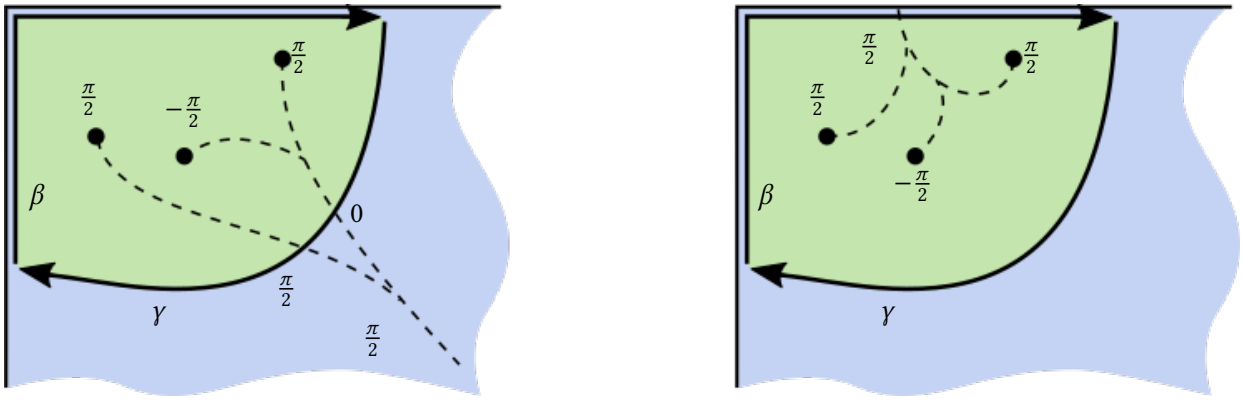


Figure 5.14: Example of the holonomy-aware extra path computation. Left: a tree of cones with computed ρ-values is shown as black dashed lines. Path γ from boundary to boundary, crossing two tree branches, has a holonomy value $\sum_\gamma \rho = \pi/2$. This path is closed along the boundary by β (with $\sum_\beta \rho = 0$), forming $m = 3$ corners. As $\sum_{\gamma+\beta} \rho = \pi/2$ and $m = 3$ conforms with Gauss-Bonnet (5.1), the path γ is admissible. Right: to illustrate that the tree of cones can be chosen arbitrarily, here the same situation is depicted with a different tree. We have $\sum_\gamma \rho = 0$ and $\sum_\beta \rho = \pi/2$, thus again $\sum_{\gamma+\beta} \rho = \pi/2$.

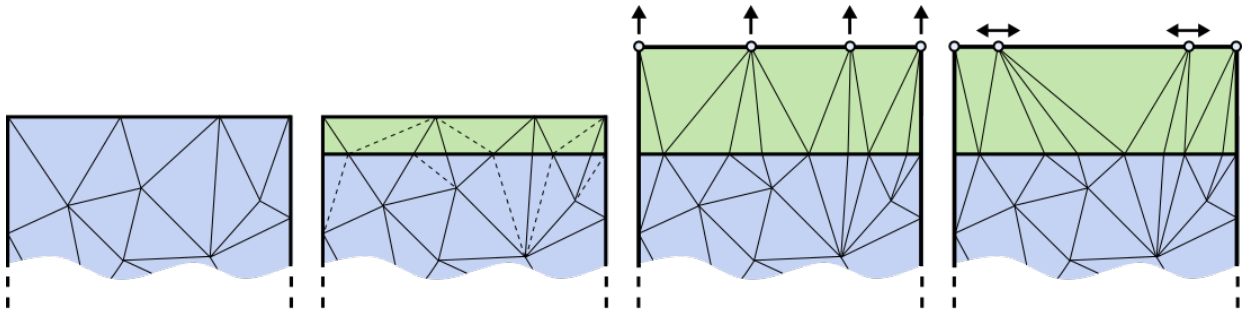


Figure 5.15: a) Mesh near a segment (top) to be padded. b) The strip to be stretched (green) is formed by inserting a straight line into the triangulation (by splitting edges at the intersections), so close to the segment that no vertex is contained. c) The strip is stretched outwards by displacing the vertices that lie on the segment by the desired padding width. d) The vertices on the segment are translated laterally according to ϕ for pointwise seamlessness.

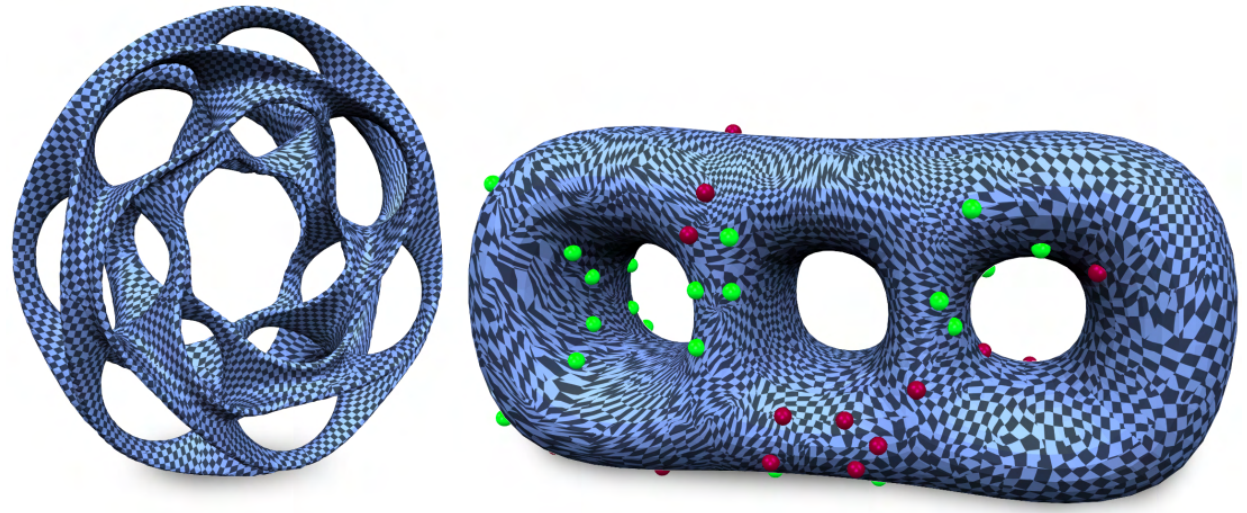


Figure 5.16: Left: example map generated on a topologically complex surface. Right: Example map generated with geometrically non-meaningful cone prescription (here: 50 randomly distributed cones of curvatures π and $-\pi$) to illustrate the method's robustness.

Figure 5.17: Visualization of a variety of locally injective seamless parametrizations obtained using our method. Note that the cut is visible in the checkerboard texture because the seamless parametrization is not a quantized seamless parametrization.

Figure 5.18: A locally injective seamless map generated on an 80-torus.

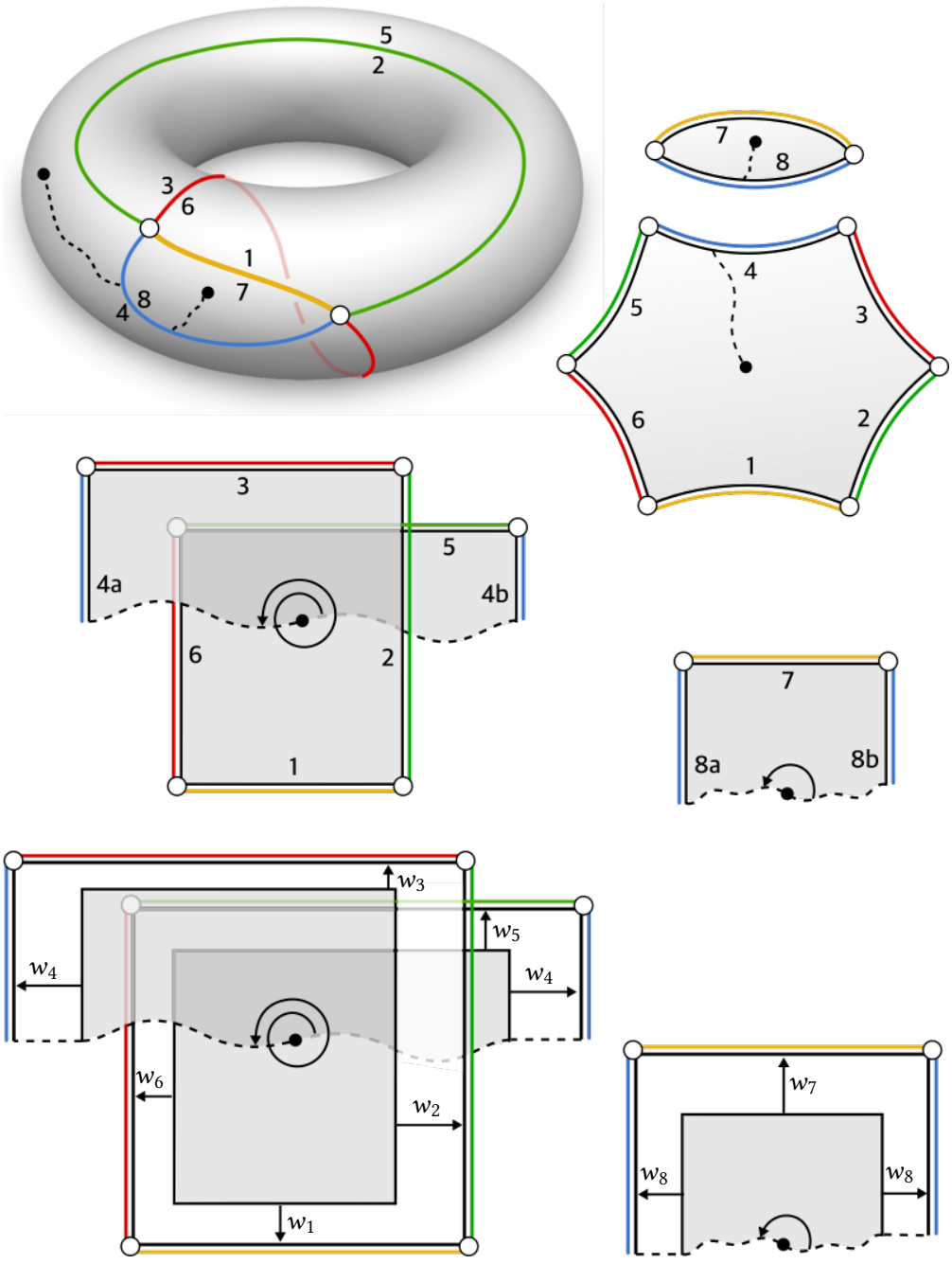


Figure 5.19: Top left: genus 1 surface with cut graph consisting of 4 branches (yellow, green, red, blue). The cut graph cuts the surface into two components with 2 and 6 corners, respectively, i.e., with a total of 8 boundary segments (two corresponding to each branch). Top right: schematic depiction of the two components under a cone metric with rectilinear boundary consisting of straight segments (here shown as curved arcs) meeting at right angles. Middle left/right: planar flattening of the two components implied by the metric (after cutting to cones – dashed). The numbering of segments is used to set up the system for padding widths w_i. Bottom left/right: the padded flattening (padding, indicated by arrows, in white).

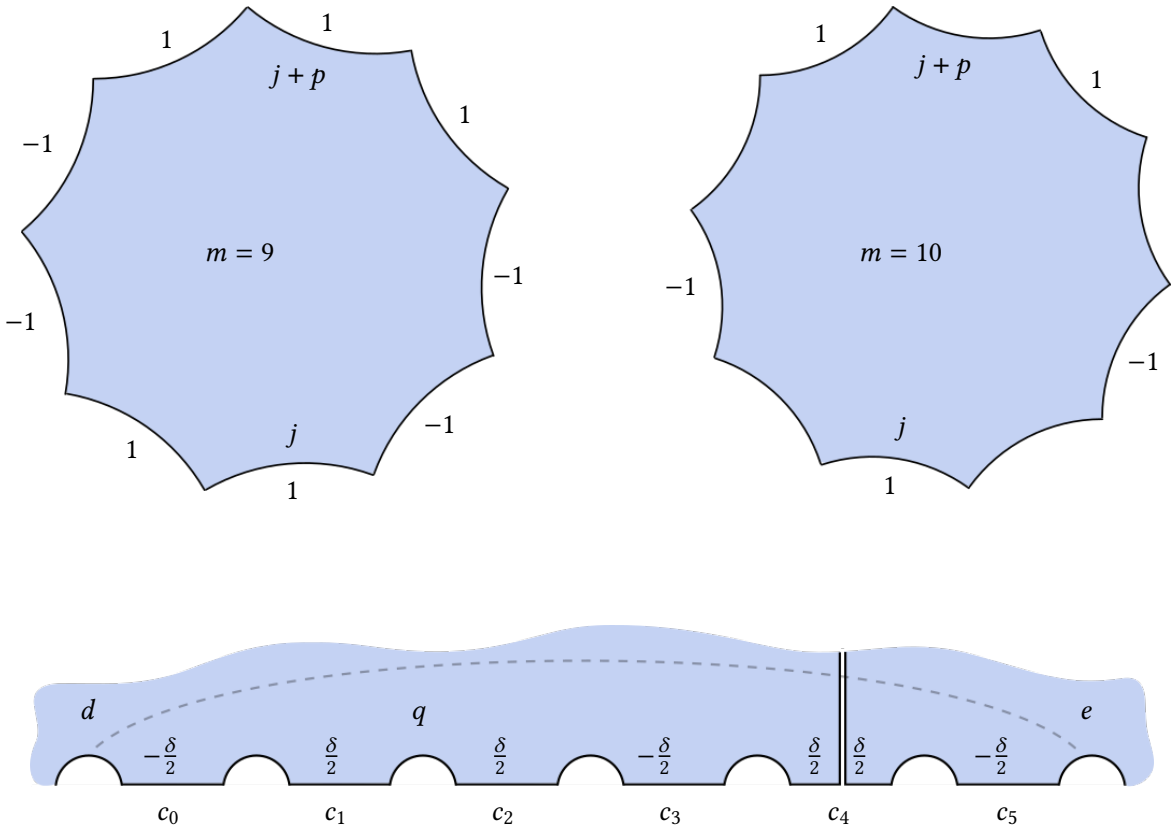


Figure 5.20: Illustration of a hole segment q between two segments of an odd-couple d–e (here with 5 hole segments between them). At c_4 an exemplary extra path connection to the hole chain is depicted.

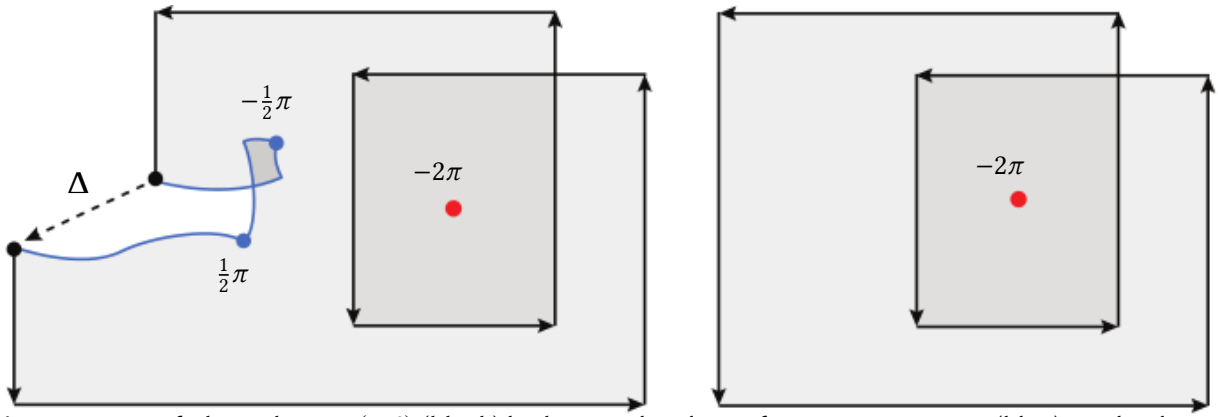


Figure 5.21: Left: boundary $\partial F(M^c)$ (black) laid out in the plane after cutting to cones (blue). Red indicates a cone with $k_i = 8$, i.e., curvature $\hat{\Theta}_i = -2\pi$ (parametric angle 4π) for which a cut is superfluous. Right: The segment gap Δ vanishes if all cones are fourfold, thus $\partial F(M^c)$ is a rectilinear polygon.

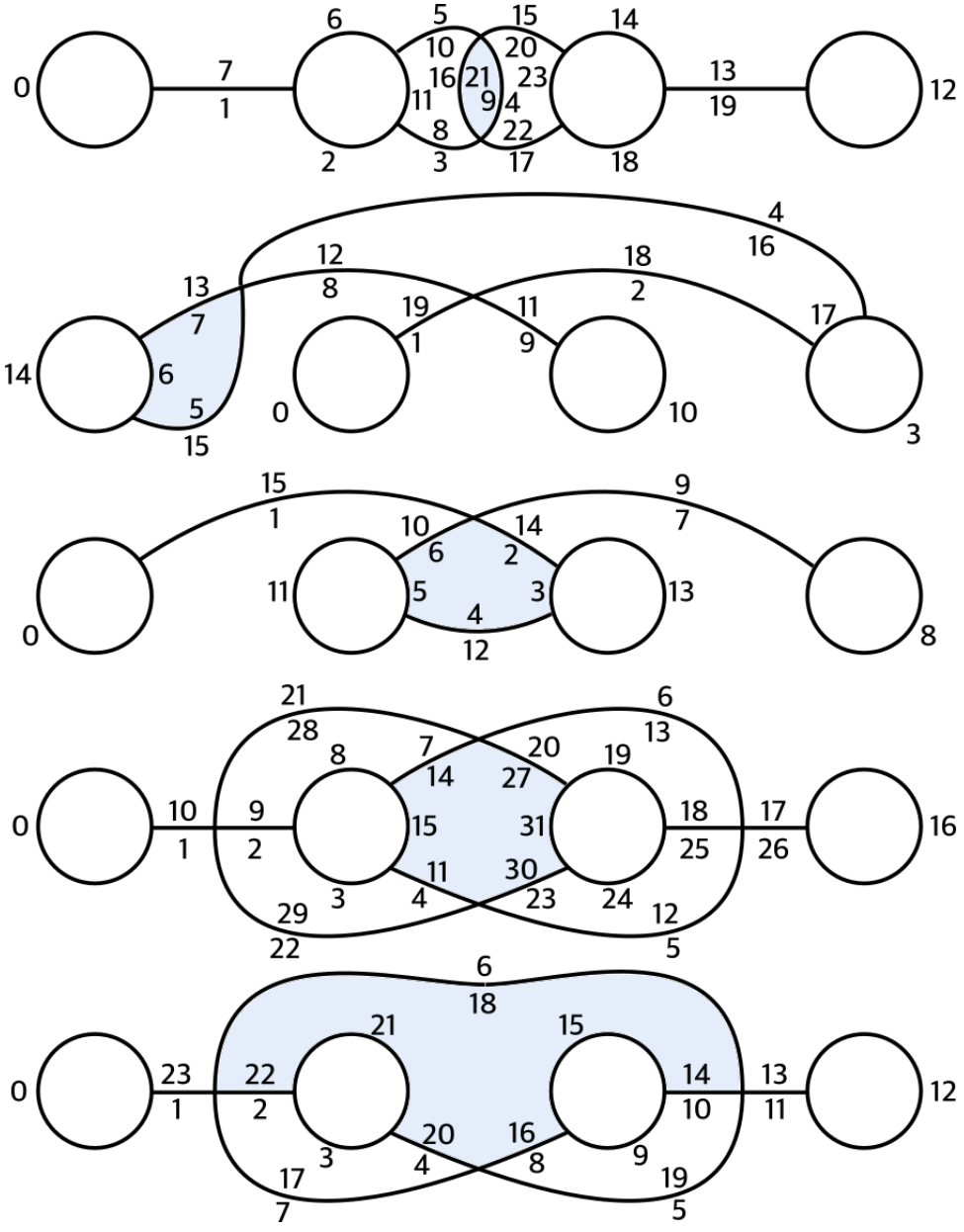


Figure 5.22: Special cut graph patterns to be used to guarantee equalizability for genus 2 surfaces, depending on whether a subset of cones compatible with a region (shaded) with 2, 3, 5, 6, or 7 corners is present.

6 Global Parameterization from Prescribed Holonomy Signatures

6.1 Introduction

Seamless surface parametrization is one of the most common approaches to constructing seamless texture atlases, conforming surface quadrangulations, and high-order (spline or subdivision) approximations to surface data. A chart-based parametrization is called seamless if it satisfies certain conditions on its transitions between charts or across cuts.

In particular, a seamless parametrization of a discrete surface defines a metric, i.e., an edge length assignment on a mesh, that is intrinsically flat almost everywhere, i.e. angles around vertices sum to 2π, except at a (often small) set of *cone vertices* with an angle deficit (or excess) of some multiple of $\frac{\pi}{2}$. More generally, the *holonomy angle* for any closed loop on the surface is a multiple of $\frac{\pi}{2}$. The holonomy angle is the angle between the first and last edge when laying out a closed chain of mesh triangles in the plane according to the metric (fig. 6.3, cf. [Bright et al. 2017; Crane et al. 2010]). Informally, this holonomy condition on a parametrization's metric ensures that parametric lines continue seamlessly across cuts, although, e.g., a u-parametric line may become a v-parametric line.

While the set of closed triangle chains on a discrete surface is infinite, all their holonomy angles are actually defined by a set of angles on a finite basis, the *holonomy signature*, which we

define more precisely below. In essence, loops around individual vertices capture all *local* aspects of holonomy, while (in case of non-trivial topology, genus > 0) a system of non-contractible loops captures the additional *global* aspects.

Holonomy Control To clarify the importance of parametrization topology defined by the holonomy signature, consider quad meshes or quad layouts obtained from (constrained classes of) seamless parametrizations by tracing a grid of parametric lines on the surface. The cones become the extraordinary vertices, where $n \neq 4$ quads meet. The holonomy angles determine how many quads meet at such extraordinary vertices, and more generally, how many turns the edges of the quads make along any closed curve on the surface, e.g., a feature line. As a consequence, controlling the parametrization's topology in the form of its holonomy angles is critical for obtaining a high-quality parametrization with intended behavior.

In many approaches to seamless parametrization, the target topology is provided as input, e.g., it is derived from a given cross-field or partially or completely specified by the user. At the same time, as we discuss in detail in section 2.1, no existing general method guarantees that the target topology is fully respected, although significant progress was made towards this goal.

Existence Moreover, to the best of our knowledge, the answer to the following question is not known:

> *For which holonomy signatures, seamless parametrizations with corresponding topology exist?*

Partially, this question was answered in [Jucovič and Trenkler 1973], and more specifically in [Campen et al. 2019], where angles at cones, but not complete signatures (including global aspects), were considered. In this paper, we resolve the question of existence for a broad class of signatures, subject to only a mild condition.

Remarkably, it turns out that for surfaces of genus $\neq 1$, there is a seamless parametrization

for *any* holonomy signature (e.g. implied by a cross-field) under this condition. For genus 1, we show that in this class the one known example of holonomy signatures for which there is no seamless parametrization (signatures with exactly two cones, with angles $3\pi/2$ and $5\pi/2$) is the only one.

For the condition to be satisfied, it is already sufficient (but not necessary) to have one cone with angle deficit $+\frac{\pi}{2}$ or $-\frac{\pi}{2}$ in the signature (corresponding to at least one valence 3 or valence 5 extraordinary quad vertex). This is essentially always satisfied for holonomy signatures implied by cross-fields optimized for smoothness or curvature alignment [Vaxman et al. 2016].

Contribution We describe an algorithm for the construction of seamless parametrizations with full control over holonomy. It extends the construction of [Campen et al. 2019] (referred to as Seamless Padding (SP) in the following) which provides control only over local holonomy aspects (i.e. cone angles). Our contribution includes:

- An existence result for seamless parametrizations with given holonomy signature, indicating a remarkably small topological gap between cross-fields and parametrizations;
- An algorithm for, given a holonomy signature, constructing an alternative system of loops on which the equivalent holonomy signature has arbitrary desired angles;
- A variant of the SP method that, based on the above, builds a valid seamless parametrization with prescribed holonomy.

We note that the topology of cross-fields—which are often used to guide the computation of seamless parametrizations—can be controlled very flexibly and precisely using existing discrete construction algorithms. In fact, one can easily construct a cross-field with any given turning number signature (the field analogue of a holonomy signature, cf. section 6.3.1) by solving a linear system of equations [Crane et al. 2010]. The ability to near-universally match this signature, provided by our method, means that this possibility of precise topology control extends to

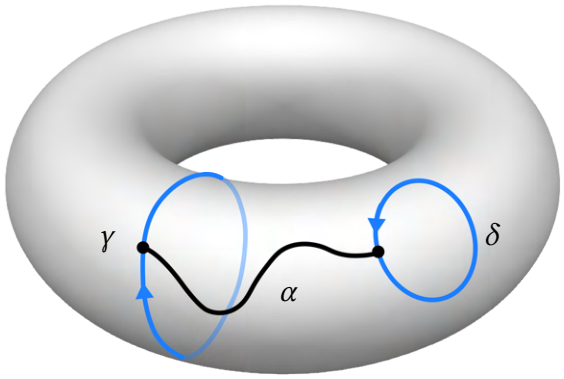


Figure 6.1: Illustration for Prop. 13 concerning quasi-additivity of holonomy numbers on loops.

parametrizations.

6.2 Existence of Seamless Parametrizations

Before discussing the algorithmic details, let us settle the question of existence of seamless parametrizations for prescribed holonomy signatures. In particular, this will allow us to guarantee that the above mentioned rerouting can actually be performed as needed.

While any choice of homology basis loops will yield a holonomy signature, our method relies on bases whose loops' holonomy numbers are some specific values. As a first step towards achieving this, the following proposition gives us a simple way to "add" together two loops so that the holonomy number of the new loop is determined by the holonomy numbers of the constituent loops.

Proposition 13 (Quasi-Additivity)**.** *Suppose γ and δ are two non-intersecting simple oriented loops and α is a path from the right side of γ to the right side of δ that only intersects these loops at its endpoints and does not contain any cones (see fig. 6.1). Then (provided the mesh is suitably refined) there is a simple loop γ_0—which can be made arbitrarily close to δ, γ, and α—such that*

$$k^F_{\gamma_0} = k^F_\gamma + k^F_\delta - 1$$

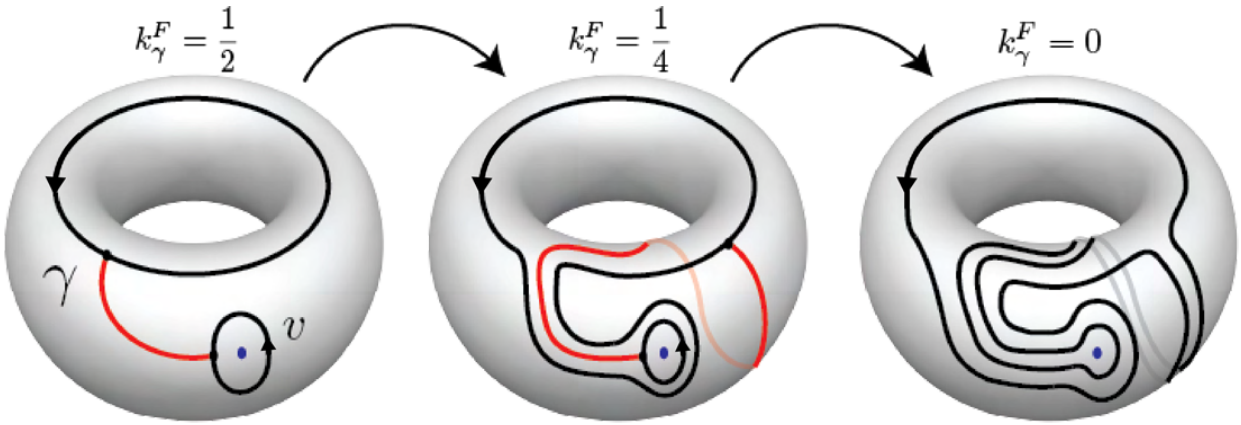


Figure 6.2: Rerouting (ccw, twice in a row) of a loop around a cone of index $\frac{1}{4}$.

If α is a path from the left side of γ to the left side of δ we have nearly the same result, but the holonomy number of γ_0 is instead given by

$$k_{\gamma_0}^F = k_\gamma^F + k_\delta^F + 1$$

A proof and an illustration can be found in section 6.9.

Rerouting around a Cone In particular, for a cone v_i, provided there is a path from the right side of γ to v_i^*, the above proposition tells us there is a loop γ_0 such that $k_{\gamma_0}^F = k_\gamma^F - I_{v_i}^F$. We refer to the construction of the latter loop as *rerouting γ around v_0* (with a counterclockwise orientation). On the other hand, if there is a path from the left hand side of γ to v_i^* (the dual facet of vertex v_i), the above proposition gives us a loop γ_0 such that $k_{\gamma_0}^F = k_\gamma^F + I_{v_i}^F$, which we refer to as rerouting γ around v_i with a clockwise orientation. Moreover, it is clear from the construction of these loops that γ_0 is homotopic to γ on M, although the homotopy will necessarily cross the cone as otherwise the holonomy numbers of the two loops would be the same. fig. 6.2 shows an example of rerouting a loop around a cone of index $\frac{1}{4}$ twice, so as to yield a loop whose holonomy number differs by $\frac{1}{2}$.

These observations lead to the following key proposition.

Proposition 14. *Let $H = \{\gamma_1, ..., \gamma_{2g}\}$ a basis of loops for M that cuts M into a topological disk, and let $v_1, ..., v_m$ be vertices of M. Also, let $k_1, ..., k_{2g}, I_1, ..., I_m \in \frac{1}{4}\mathbb{Z}$ and assume $\gcd(I_1, ..., I_m) = \frac{1}{4}$. Then there is another basis of loops $\delta_1, ..., \delta_{2g}$ that cuts M into a disk such that $k^F_{\delta_i} = k^F_{\gamma_i} + k_i$, $i = 1, ..., 2g$, for any seamless parametrization F that has cones with indices $I^F_{v_j} = I_j$ at the vertices $v_1, ..., v_m$.*

For a constructive proof see section 6.10. Conceptually, we can reroute the loops γ_i one-by-one around suitable subsets of the cones in a manner that preserves the topology of H, such that their holonomy numbers change exactly by the desired values k_i.

For the purpose of our method, this result means that we can start from a cut graph formed by the union of $2g$ loops $\gamma_1, ..., \gamma_{2g}$, and modify these loops using an appropriate choice of integers $k_1, ..., k_{2g}$ to yield loops $\delta_1, ..., \delta_{2g}$ instead, with any holonomy numbers we want, forming an equivalent signature—under the only condition that $\gcd(I_1, ..., I_m) = \frac{1}{4}$. This ability is sufficient for the method presented in the following to construct a seamless parametrization with the desired holonomy, which constructively shows existence, under the above condition.

GCD-Condition This condition is obviously satisfied as soon as there is even just one cone of index $\pm\frac{1}{4}$ among all prescribed cones. But this (practically very mild assumption) is not even necessary; even if all indices are of higher magnitude, they may have a greatest common divisor of $\frac{1}{4}$. If the gcd is indeed larger than $\frac{1}{4}$ (a potentially realistic scenario is one with indices restricted to multiples of $\frac{1}{2}$), note that while not all holonomy numbers can be achieved by rerouting, it may still be possible to achieve those desired.

6.3 Holonomy Signature

We consider a closed orientable manifold mesh M of genus g and a cut graph G on M that cuts M to one or more topological disks. We let M^c denote the resulting cut mesh, which has a canonical map $\pi : M^c \to M$ that is the identity on the interior and maps exactly two boundary edges in M^c

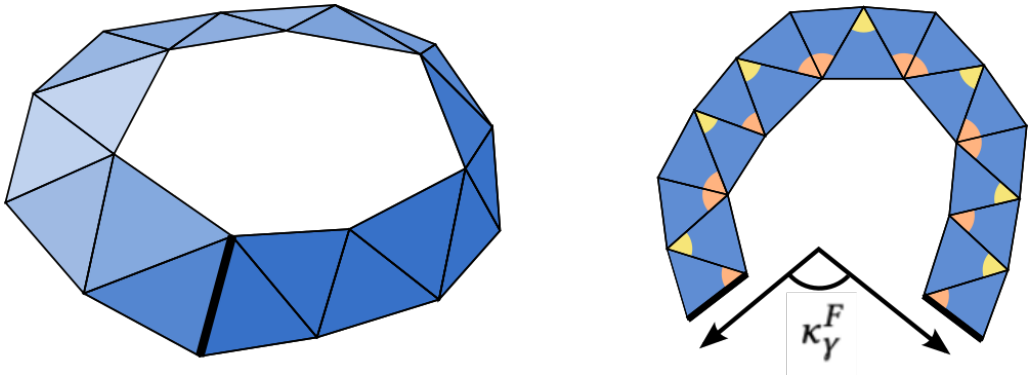


Figure 6.3: The holonomy angle κ_γ^F (def.2) of a dual loop (cyclic triangle strip) under a metric F is the sum of signed inner angles (yellow and orange). Up to multiples of 2π (if the loop makes multiple turns) this corresponds to the angle between first and last edge when laying out the strip in the plane.

to each edge in $G \subset M$. We call two edges e, e' in the boundary of M^c *mates* if $\pi(e) = \pi(e')$. We also define a *loop* as an oriented closed walk of facets (or dual vertices) of M and a *simple loop* as an oriented cycle of facets (or dual vertices).

Definition 1 (Seamless Parametrization)**.** *A discrete seamless parametrization, as in [Myles and Zorin 2013], is a continuous piecewise linear, locally injective map $F : M^c \longrightarrow \mathbb{R}^2$ such that for any boundary edge e with mate e', there is a rigid transformation $\sigma_e(x) = R_e x + t_e$, where R_e is a rotation by an integer multiple of $\frac{\pi}{2}$, that maps $F(e)$ to $F(e')$, i.e. $\sigma_e(F(e)) = F(e')$.*

A seamless parametrization naturally induces a discrete metric $E \longrightarrow \mathbb{R}^{>0}$ on M^c by letting the length of an edge e of M^c be the length of $F(e) \subset \mathbb{R}^2$. Since mated edges e, e' in the boundary of M^c are related by a rigid transformation, $F(e)$ and $F(e')$ have the same length, so this metric extends to a well-defined metric on M. Moreover, the metric on M is flat except at isolated vertices $C = \{v_1, ..., v_m\}$ in G, i.e. in the boundary of M^c, called *cones*.

Definition 2 (Holonomy Angle)**.** *For a loop γ, the holonomy angle (or discrete geodesic curvature [Crane et al. 2010]) of γ under a seamless parametrization F is*

$$\kappa_\gamma^F = \sum_{f^* \in \gamma} \alpha_\gamma(f^*),$$

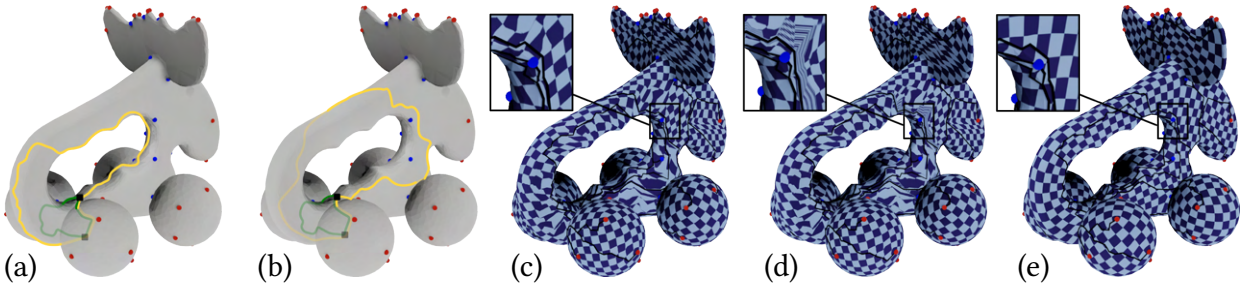


Figure 6.4: Algorithm overview: (a) Example input signature loops (yellow and green) and cones (red and blue). (b) Loops of an equivalent signature obtained by strategically modifying this input; notice that the yellow loop takes a different path between the cones. (c) Conformal parametrization respecting the prescribed cones and aligned with the cut graph that is formed by the loops; due to this alignment, it has a specific holonomy pattern along the loops. (d) The map is modified by parametric padding to make it seamless while preserving its holonomy properties. (e) Finally, the map can be continuously optimized for low distortion and possibly cross field alignment, naturally within its topological class.

where f^* is the vertex dual to facet f, $\alpha_\gamma(f^*)$ is the signed angle of $F(f)$ at the vertex of f incident to the preceding and succeeding facets in the loop. The sign is positive (negative) for vertices on left (right) hand side of the loop. See fig. 6.3 for an illustration.

Let v^* denote the dual facet of vertex v, and ∂v^* the cycle of this dual facet's dual vertices. In other words, ∂v^* is the loop around the single vertex v.

Definition 3 (Holonomy Number)**.** *For a loop γ, we define the* holonomy number *as $k^F_\gamma = \kappa^F_\gamma / 2\pi$. In the case of a vertex-loop, $\gamma = \partial v^*$, we additionally define the* index *of the vertex as $I^F_v = 1 - k^F_{\partial v^*}$.*

Since the images of edges in the boundary of M^c under F are related by rotation angles that are integer multiples of $\frac{\pi}{2}$, the holonomy numbers are always integer multiples of $\frac{1}{4}$.

Definition 4 (Holonomy Signature)**.** *We call the holonomy numbers for a choice of homology basis loops $\gamma_1, ..., \gamma_{2g+m}$ of $M \setminus C$ the holonomy signature of F. A natural choice of basis is a homology basis $\gamma_1, ..., \gamma_{2g}$ of M together with cone vertex loops $\partial v^*_1, ..., \partial v^*_m$. The loops are referred to as signature loops.*

A homology basis of M can, for instance, be chosen as a so-called *system of loops* [Erickson and Whittlesey 2005].

Importantly, such a finite holonomy signature completely captures the holonomy number of *any* loop: First, since the metric is flat except at cones, any two loops homotopic in $M \setminus C$ (i.e., loops that can be continuously deformed into each other within M without crossing a cone vertex) have the same holonomy number (cf. Prop. 1 in [Myles and Zorin 2012]). Second, given an arbitrary loop γ and a homology basis of $M \setminus C$, there is (by the nature of a homology basis [Hatcher 2002]) a surface $\bar{M}$ so that its boundary is composed of γ and some combination of the basis loops. The Gauss-Bonnet theorem then gives a formula for the holonomy number of this boundary in terms of the Euler characteristic of $\bar{M}$. Thus, the holonomy number of γ is determined by the holonomy numbers of the loops of a homology basis (cf. Prop. 2 in [Myles and Zorin 2012]). fig. 6.4 (a) shows an example of signature loops: the green and yellow loops form a homology basis of M, while the small cone vertex loops are visualized as red and blue dots at the respective vertices.

Note that the holonomy signature is not unique, neither its loops nor its numbers (fig. 6.5). For a fixed seamless parametrization F, a different choice of signature loops will lead to different associated holonomy numbers—even though the same parametrization topology is represented. Such holonomy signatures are called *equivalent*.

6.3.1 Relation to Cross-Fields

Seamless parametrizations are often employed in conjunction with cross-fields, most importantly when parametrizations are built and optimized for directional alignment with such a field. In such cases it is important for field topology and parametrization topology to match. In this context, there is a close connection between the holonomy of a seamless parametrization and the turning numbers of a cross-field.

For a smooth surface S, a cross-field $\vec{d}$ is a differentiable mapping of four tangent vectors to each point $p \in S$ (except at isolated singularities) that are invariant to rotation by $\pi/2$ around the normal $\hat{n}_p$ to S at p. Given such a field and a loop γ on the surface S, the field will make some number of rotations along this loop, and due to the rotational symmetry of the field these turning

numbers T_γ can be integer multiples of $\frac{1}{4}$.

Moreover, as shown in [Ray et al. 2008], there is a discrete analogue of cross-fields and turning numbers for triangle meshes, and turning numbers along loops satisfy a theorem analogous to the Poincaré-Hopf theorem for vector fields that states $T_{\partial S} = -\chi(S)$. This implies that the holonomy number of the boundary of a flat surface, which does not contain any cone, is the same as the turning number of a singularity-free cross-field along that boundary. Hence, if turning numbers of a cross field agree with holonomy numbers on a set of signature loops then they will agree for *any* loop on the surface. Consequently, by taking as our desired holonomy numbers the turning numbers of a given cross-field on a homology basis of $M \setminus C$, where C is the set of the cross-field's singularities, the seamless parametrization is fully constrained to topologically match the input cross-field—in terms of local (cone indices) as well as global behavior.

6.4 Approach Overview

Given a holonomy signature (or a cross-field implying a holonomy signature, cf. section 6.3.1), consisting of loops associated with a holonomy number each, on a surface M, our goal is to construct a valid seamless parametrization F for M that respects this signature. In section 6.2 we discuss the question for which signatures this is actually feasible.

Key Idea We show in section 6.2 that, given a holonomy signature, we can find an equivalent signature (by exchanging or modifying the signature loops) such that the associated holonomy numbers assume almost any desired values. Essentially, we are exploiting the above mentioned non-uniqueness of the signature (cf. fig. 6.5). We make use of this algorithmically in section 6.5 in the following way: The SP method [Campen et al. 2019] enables constructing a seamless parametrization that has prescribed local holonomy (i.e. cones), but it lacks the ability to prescribe holonomy globally. However, its result has not a random but a fixed holonomy pattern

along the cut graph that is used in the construction. We therefore modify the $2g$ global signature loops such that their union forms a cut graph and such that their corresponding holonomy numbers in an equivalent signature match exactly the fixed pattern that SP will produce. fig. 6.4 illustrates the main steps of our construction process.

Cut Graph The seamless parametrization construction by SP relies on using cut graphs with certain structural restrictions, so-called hole-chains (or modifications thereof). fig. 6.6 shows an example, details follow in section 6.5.1. Let G be such a cut graph. Let H be a *system of loops* of M, i.e., H is a homology basis and cuts M into a topological disk. In particular, as G cuts M into a topological disk, H can be chosen such that the (non-disjoint) union of its loops equals G (section 6.5.2).

Fixed-Holonomy Parametrization By construction, SP will yield certain predetermined holonomy numbers along G, thus on these loops H, regardless of the loops' geometry. More concretely, each branch point of the hole-chain G has four incident cut segments, conceptually forming a cross. When following a loop through G, at such a branch point it may therefore take a left-turn, a right-turn, or continue straight. The generated parametrization's holonomy number along any loop in G is simply the (signed) difference between its number of right-turns and its number of left-turns (times $\frac{1}{4}$). This is due to the SP-parametrization being aligned to all segments of G, i.e., they are geodesic under this metric, and the corners between segments at branch points form right angles under this metric.

Cut Graph Rerouting Given target holonomy numbers on the loops of H (e.g., derived from an input cross-field), by Prop. 14 we can modify the loops of H, yielding H', such that their target holonomy numbers equal their left-right-turn balance. Conceptually, this rerouting of loops is described in section 6.2; in section 6.5.3 we describe algorithms that practically implement it. As this rerouting does not alter the loops' intersections, i.e., it preserves the branch points of G and

the loops' left-right-turns, the union of loops from H' still forms a hole-chain, which we can then use as prescribed cut graph for the parametrization construction.

Holonomy Soft-Guidance In order to yield parametrizations that are not just topologically correct but also (already initially, before distortion optimization) of reasonable geometric quality, we aim to reduce the need for cut graph rerouting as much as possible. To this end we construct the individual paths that the initial hole-chain G is made of in a holonomy-guided (e.g. cross-field guided) manner (sections 6.5.1.1 and 6.5.1.2). This promotes hole-chains that largely have the desired holonomy properties right away.

6.4.1 Algorithmic Outline

Our method's overall algorithmic pipeline can be outlined as follows:

1. Construct Cut Graph
    - Initial field-guided hole-chain G (section 6.5.1)
    - Extract loop basis H of G (section 6.5.2)
    - Reroute $H \rightarrow H'$, yield G' (section 6.5.3)
2. Construct Seamless Parametrization
    - Construct G'-aligned mapping $f : M^c \rightarrow \Omega$ (section 6.6.1)
    - Pad f to yield seamless map $f' : M^c \rightarrow \Omega'$ (section 6.6.2)
    - Optimize f' and Ω', maintaining seamlessness (section 6.6.3)

6.5 Holonomy-Constrained Cut Graph

The cut graph with particular holonomy pattern is built in three steps. We start by constructing a hole-chain G; in deviation from the algorithm described for this purpose in [Campen et al. 2019]

we employ soft-guidance by a given input cross-field already in this step. Afterwards a holonomy basis of loops H is extracted from G, and its associated target holonomy numbers are derived from the cross-field. Finally, these loops are rerouted where necessary, i.e., where soft-guidance did not yield exactly those holonomy numbers we require for the subsequent stage.

6.5.1 Field-Guided Hole-Chain

A hole-chain G on M is built out of g loops (non-contractible, non-separating, non-homotopic) and $2g - 1$ connecting paths. Intuitively, cutting the surface by the g loops yields a topological sphere with $2g$ holes, and the $2g - 1$ paths connect these in a chain-like manner, further cutting the surface to a topological sphere with one hole, i.e. a disk. fig. 6.6 shows an example with 4 loops (circular, inside the tunnels) and 7 connectors. We here describe how to construct these loops and connectors guided by a given cross-field.

Remark: For certain special cases (genus ≤ 2) a slightly modified hole-chain structure needs to be chosen. This is done exactly as in the SP method. Likewise, an extra connector path possibly needs to be added; this occurs after rerouting (section 6.5.3).

6.5.1.1 Field-Guided Loops

We construct g non-contractible, non-separating, non-homotopic loops on $M \setminus C$ following the algorithm of [Diaz-Gutierrez et al. 2009]. To promote cross-field alignment (thus turning number zero along the loop) we employ the field-alignment metric of [Campen et al. 2012] in this process. This in particular means that the loop construction is performed on M_4, a four-sheeted covering of M, owing to the four different directions a cross-field specifies per point.

A loop resulting from this, while simple (i.e. intersection-free) on M_4 by construction, may in some cases be self-intersecting when projected down onto M. Conceptually, it may pass "over" itself on a different sheet of M_4, which corresponds to an actual crossing on M. In such a case we

fall back to a non-guided construction of a replacement loop directly on M.

6.5.1.2 Field-Guided Connectors

Connector paths between the loops are selected in a Hamiltonian path manner as in the SP method. By contrast, however, we do not build these from simple shortest paths but again in a field-guided manner. As in the loop construction in section 6.5.1.1 we use an anisotropic metric and perform the path search on M_4. As the above loops have an embedding in M_4 by construction, we know on which sheet of M_4 to start and end the search, respectively: one sheet lower or higher than the respective loop, as a connector will be orthogonal (rather than parallel) to its two incident loops under the final parametrization. Again, should a self-intersecting connector path occur, we fall back to a shortest path computed on M.

Remark: Loops constructed by the fallback method (if any) have no native embedding on M_4. We locally (at the intended connector start or end point) assign them to the sheet on which the field direction best fits the loop's tangent, so as to have a reasonable setup for the computation of incident connectors. In any case, however, let us remind that the worst possible outcome of a locally suboptimal choice is a hole-chain that requires some more rerouting—structural correctness is not at stake in this soft-guided approach.

The resulting loops and connectors are embedded in edges of M and together form the discrete cut graph G.

6.5.2 Homology Basis Extraction

We construct a homology basis H of M in the form of $2g$ loops contained in the cut graph G. To this end we compute a spanning tree T in G. The remainder $G \backslash T$ consists of $2g$ edges, called bridges [Erickson and Whittlesey 2005]. For each bridge, its union with the two paths from its

incident vertices to the root of T is a loop, and these $2g$ loops form a homology basis, and more specifically a system of loops.

Note that these loops may coincide partially. Each of the $2g$ bridge edges, however, is part of exactly one of these loops only. By rerouting the segments of G that contain these bridge edges (called bridge segments) we are therefore able to *individually* alter the holonomy number or turning number of each of these $2g$ loops with respect to a given field. Effectively, the bridge segments are the places where the conceptual α-path from the proof of Prop. 14 can be attached without intersecting any other basis loops. Ultimately, a modified cut graph G' with the desired holonomy number for each basis loop can be obtained in this way, as detailed in the following.

6.5.3 Segment Rerouting

For each loop of H we count its number of left turns m_l and right turns m_r (in ccw sense). The holonomy number along this loop in the parametrization we will construct will be $\frac{1}{4}(m_l - m_r)$ (cf. fig. 6.6). If its target holonomy number is t (e.g., the cross-field turning number along this loop), we need to reroute this loop such that this number changes by $k = \frac{1}{4}(m_l - m_r) - t$.

This is performed by rerouting the loop's bridge segment, which we tackle in a two-tier manner. We first attempt to find a replacement path for the segment by an efficient field-guided method, detailed in section 6.5.3.1. As this method is not guaranteed to yield a *simple* path (which, however, is needed), where necessary a guaranteed (but less geometry aware) fallback strategy is employed, as described in section 6.5.3.2. Note that the resulting loops are not unique; many different equivalent signatures exhibit the desired holonomy numbers. Due to equivalence, however, the final parametrization's topology is not affected by this (see fig. 6.7).

Remark: Optionally, we may perform a pre-rerouting step for the field-guided loops from section 6.5.1.1 already before moving on to the connector computation. This is possible because these loops' target holonomy is known to be zero, regardless of how the connectors will interact

with them. This pre-rerouting is not necessary for correctness, but empirically it reduces the total amount of rerouting required. As the g loops do not yet form a complete cut graph that cuts the surface to a disk, one needs to take one additional precaution, though, so as to ensure that a loop is rerouted homotopically. Namely, we cut M minus the loops to a disk by additional temporary cuts (using the method of [Erickson and Whittlesey 2005]) and perform rerouting within this disk.

6.5.3.1 Holonomy-Aware Dijkstra

Given a bridge segment ℓ of G, supposed to be rerouted such that the holonomy number of its unique containing loop γ from H changes by k, we employ a holonomy-constrained Dijkstra's algorithm, as described in [Campen et al. 2019, §5.1]. This entails the following. We compute a spanning tree of M^c (cut by the current G), rooted on ℓ. Indices of cone vertices are propagated through this tree towards the root, and tree edges are marked with the sum of indices propagated through them. By then keeping track of the sum of these values of edges crossed during Dijkstra's shortest path algorithm (applied to the dual mesh), we can read off the index sum of cones enclosed between a Dijkstra path ℓ' and bridge segment ℓ. The algorithm terminates when a path enclosing the desired index sum (which determines the change to the holonomy number of γ) is found.

Similar to the path construction on M_4 described in section 6.5.1, this holonomy-constrained path search effectively occurs on an (in this case infinite) cover of M (akin to the universal cover of $M \setminus C$). Consequently, a non-simple path (in M) can be the result in some cases. The following guaranteed fallback strategy takes care of such cases.

6.5.3.2 Fallback Strategy

By following the rerouting construction used in the proof of Prop. 14, a proper simple replacement path for a bridge segment ℓ can safely be found. Let v be a vertex on a bridge segment ℓ whose

loop's holonomy number needs to be increased by k.

Assume for a moment that there is a cone vertex v^* with $|I(v^*)| \leq k$. Let α be the shortest path from cone v^* to v—either meeting ℓ from the right if I and k have opposite sign, or from the left in the case of equal sign. Among all suitable cones, we choose the one for which the path α is shortest, so as to reduce the amount of modification.

Let the two vertices on ℓ which are directly adjacent to v be v^- and v^+. Closely following the conceptual rerouting from fig. 6.10 we remove the edges v^-v and vv^+ from ℓ and replace them by the path from v^- to v^+ tightly along α and around v^*. Where necessary we split edges of M to make room so that this path does not touch any other part of G.

This changes the loop's holonomy number by $I(v^*)$ or $-I(v^*)$, depending on which side of ℓ the path α connects to. This procedure can be repeated as long as the remaining difference $k \leftarrow k \pm I(v^*)$ is not zero yet.

While this strategy proved sufficient in all practical test cases (cf. section 6.7) (and indeed is guaranteed to work if there is at least one cone of index $\pm\frac{1}{4}$), in general a greedy selection of reroute-cones v^* in this greedy manner is insufficient. Instead, let V be a multiset of cones such that its index sum is k. Under the GCD-condition (Prop. 6.2), V exists, as also exploited in the proof of Prop. 14. By selecting the cones from V as v^* in the above one after the other (also considering multiplicity), the desired result is achieved. A suitable multiset V, i.e. a subset of cones and multiplicities, is easily computed using the Extended Euclidean Algorithm. The incorporation of distances also in this general case would enable smaller rerouting modification, but given the practical irrelevance of this multiset-case, the effort would hardly pay off.

Remark: In practice we tentatively perform the fallback-rerouting starting from multiple root vertices v (sampled equidistantly on the bridge segment; we use 10 samples in our experiments) and retain the result of shortest length to reduce the complexity of the final cut graph.

6.6 Seamless Parametrization

Once the final, i.e., rerouted and possibly extended (recall the remark in section 6.5.1), cut graph G' is available, the next step is to construct a domain that is compatible with the cut surface M^c and suitable to serve as parameter domain for a seamless parametrization of M. The domain shape is derived from a conformal metric computed on M^c with prescribed cones and prescribed boundary curvature.

6.6.1 Cut Graph aligned Metric

A key role in the SP method that we build on is played by a discrete conformal metric computation on the cut mesh M^c. While the conformal metric algorithm from [Campen and Zorin 2017b] that is used in SP works adequately in most cases, it does not provide strict guarantees of convergence. The cut graph rerouting used in our method can sometimes lead to rather complex cut shapes, thus boundary shapes of M^c, implying additional metric distortion, making the problem instances particularly challenging.

Very recently, a novel algorithm for discrete metric computation with prescribed (boundary and cone) angles has been proposed [Campen et al. 2021; Gillespie et al. 2021], based on mathematical insights [Gu et al. 2018b; Springborn 2019] that guarantee convergence. We employ an implementation based on this work.

Using this algorithm we compute a discrete metric (i.e., edge lengths) for M^c, prescribing the angles of cone vertices and the geodesic curvature on the boundary, i.e., along the cut G'. Concretely, the segments of G' are constrained to be straight under the resulting metric, and the corners (at branch points of G') are constrained to be right.

6.6.2 Padding

Under the computed metric the two boundary segments of M^c corresponding to a common segment of G' are straight, their mutual angle is a multiple of $\pi/2$ (as required for seamlessness), but their lengths may differ. The SP method uses so-called *padding*, i.e., parametrically stretching out strips of the surface under the metric along the boundary segments so that the boundary segments' lengths expand to equalize the lengths of all paired segments, and this provably is always possible.

The following steps perform this padding, analogous to the original SP method from [Campen et al. 2019]:

1. Add cuts from all cones to the boundary of M^c. Make sure each boundary segment is reached in at most one point. Around each interior vertex the angles under the metric from section 6.6.1 now sum up to 2π, i.e. the cut surface is flat.

2. Lay out this flat mesh in the plane, i.e., assign (u, v)-parameters to all vertices of the mesh, e.g., in a breadth-first traversal. The global rotation is chosen such that the straight boundary segments are aligned with u or v axis directions. This yields the (non-seamless) domain Ω.

3. For each straight boundary segment, compute the amount of padding (width w_i) required to equalize parametric lengths of the paired segments, using the equation system of the SP method.

4. Along each segment, determine a parametrically rectangular strip free of cones, and make the mesh conform to this strip by inserting the strip's boundary line by splitting. Then apply a stretch transformation to the (u, v)-coordinates inside the strip, so as to shift the segment in perpendicular direction by its padding width w_i (fig. 6.8).

5. In cases where the cut graph has cut the surface into more than one disk, glue these together parametrically along pairs of boundary segments by means of rigid transformations applied to the (u, v)-coordinates to finally obtain a map F' onto a single connected domain Ω'.

6.6.3 Optimization

As a final step, we optimize the established map for reduced distortion. As objective, we employ a local cross-field (orientation and sizing) alignment energy E_A [Bommes et al. 2009b] and add (with a small factor of $s = 10^{-3}$) the symmetric Dirichlet energy E_D [Rabinovich et al. 2017b], which contributes its barrier behavior to prevent parametric inversions in the course of optimization. Linear constraints are added to preserve seamlessness. We use a projected Newton solver and use an explicit triangle inversion check in the line search [Smith and Schaefer 2015], using exact predicates, to reliably maintain local injectivity. We experimentally discovered that using an unconstrained Newton optimizer over the set of independent variables computed using a reduced row echelon form of the constraint matrix is numerically more stable than solving a KKT system at each iteration, leading to faster convergence.

We emphasize that we do not aim to address map optimality here; our focus is on constructing a topologically correct initial map, subject to further improvement geometrically.

6.7 Evaluation

We apply our method to a dataset of 3D models with cross-fields [Myles et al. 2014]. We restrict ourselves to models of genus > 0, as on topologically trivial surfaces there are no global holonomy aspects to account for. The method succeeds in generating a cut graph with exactly the needed holonomy numbers in all cases. As all cases satisfy the gcd=$\frac{1}{4}$ condition, and all crucial operations are combinatorial/discrete, the general success of this step is indeed to be expected. For each model, table 6.1 lists the number of rerouting operations that our method performed.

The construction of a seamless parametrization based on this cut then succeeds in most cases; in six, however, the initial metric distortion is very high, causing subsequent steps (padding or the simple distortion optimization approach) to get into numerical trouble. In fig. 6.9 obtained optimized seamless parametrizations for examples from the dataset are shown, matching the input cross-field by construction. table 6.2 reports the final distortion of these.

6.7.1 Comparison

To demonstrate the importance of our contribution in the context of guaranteed locally injective seamless parametrization construction, we also apply the bare SP method of [Campen et al. 2019] (which takes local holonomy (cones) but not global holonomy into account) to these models.

While SP is able to respect the singularities of the prescribed field by construction, whether or not its resulting map matches the cross-field topologically is essentially a matter of chance. If the cross-field is very smooth (as generally is the case in this data set) and the cut graph for the map is constructed from certain shortest paths, the chance of a match may be higher than that of any particular mismatch. Nevertheless, we encounter a mismatch for a large number of models—in line with the fact that, as can be seen in table 6.1, our method had to employ at least one rerouting operation in the majority of cases. In case of a mismatch, the resulting map cannot continuously be optimized to achieve reasonable alignment between map isolines and the field, as there is a topological obstacle. This can be observed in table 6.2, where the remaining final distortion is significantly higher when not employing rerouting. The difference is also illustrated in fig. 6.9. Our method, in essence by adjusting the cut graph in the described manner, ensures a topological match between the signature induced by the cross-field and the signature of the generated seamless parametrization.

6.8 Conclusion and Future Work

We have explored the relation between cross-fields and seamless surface parametrizations (and therefore quadrangulations) on a topological level. A key insight is that there are hardly any practically important obstacles to generating a seamless parametrization (or quadrangulation) that topologically matches a given cross-field. We have described a method to generate such a seamless parametrization, given an input cross-field or an abstract topological specification in form of a holonomy signature. It is based on a variation of the SP method [Campen et al. 2019], with the main difference being:

- The initial hole chain cut graph is constructed taking cross-field guidance into account.
- The hole chain is then modified by extracting a loop basis and rerouting of loop segments based on our theory.
- The generation of a cut-aligned parametrization is performed using a different, theoretically sound conformal mapping method.

From the SP method that we employ for the parametrization construction we inherit the restriction to surfaces without boundary. While there are no fundamental obstacles to adding boundary support to our rerouting procedure, padding feasibility requires additional theory in this more general context. The situation regarding support for alignment to feature curves, which is of interest in some use cases of seamless parametrizations, is very similar.

The algorithm stage described in section 6.5, in particular the holonomy-constrained cut graph generation using rerouting, relies on discrete operations and therefore is not only sound theoretically, but can be executed without the risk of numerical issues and limits in practice. The algorithm stage described in section 6.6 (initial parametrization followed by constrained optimization), by contrast, involves numerical computations, with consequent limits in practice.

While for initial parametrization a discrete approach is imaginable [Zhou et al. 2020], at least for the final distortion optimization a numerical approach is inevitable.

While we observe the choice of loops that form the initial cut graph to not affect the final result conceptually (fig. 6.7), the distortion of the initial parametrization, and therefore the numerical challenges in the final optimization, can depend strongly on this choice. By testing various random root placements for the loop construction [Diaz-Gutierrez et al. 2009] employed in section 6.5.1.1, initial parametrizations of low distortion could be found, but a more direct approach—or a more resilient final distortion optimization technique—is desirable.

The GCD-condition asserts that, for any given signature, there is an equivalent signature whose loops have any desired set of holonomy numbers. It therefore is a sufficient condition for the existence of a seamless parametrization that topologically matches a given signature. It is not necessary, though. While likely of limited practical relevance, the exploration of even tighter conditions may be interesting.

6.9 Proof of Proposition 1

Proof. We consider the case where α is a path between the right hand sides of the two loops. If the mesh is suitably refined, there is a topological disk D in the dual mesh that contains α and does not contain any cones (fig. 6.10 left). Without loss of generality, we may assume that ∂D intersects γ along a single nontrivial path β_γ and intersects δ along a similar path β_δ. We denote the endpoints of β_γ as f_1^* and f_2^* and the endpoints of β_δ as g_1^* and g_2^*. We now can define γ_0 as the simple loop given by traversing $\gamma \setminus \beta_\gamma$ (with respect to the orientation of this loop) starting at f_2^*, then traversing the component of ∂D (with boundary orientation) from f_1^* to g_2^*, then traversing $\delta \setminus \beta_\delta$, and finally by traversing ∂D from g_1^* to f_2^*. We note that we can choose D so that the boundary is arbitrarily close to α and thus so γ_0 is arbitrarily close to the original loops and path.

Since ∂D is the boundary of a topological disk that does not contain any cones, we have that

$\kappa^F_{\partial D} = 2\pi$. In the computation of $\kappa^F_{\gamma_0}$, we have that the signed angles satisfy $\alpha_{\gamma_0}(f^*) = \alpha_\gamma(f^*)$ for $f^* \in \gamma \setminus \beta_\gamma$ and $\alpha_{\partial D}(f^*) = -\alpha_\gamma(f^*)$ for $f^* \in \beta_\gamma \setminus \{f_1^*, f_2^*\}$ (fig. 6.10 right). Furthermore, since α intersects γ on the right, we must have that the angle of f_1 that corresponds to $\alpha_\gamma(f_1^*)$ is on the left hand side of γ, a different angle of f_1 corresponds to $\alpha_{\partial D}(f_1^*)$ and is on the left hand side of ∂D, and the final angle of f_1 corresponds to $\alpha_{\gamma_0}(f_1^*)$ and is on the right hand side of γ_0. Therefore, we have that

$$\alpha_\gamma(f_1^*) + \alpha_{\partial D}(f_1^*) - \alpha_{\gamma_0}(f_1^*) = \pi,$$

and by a similar analysis the same result for f_2^* is obtained. The situation is similar for δ and ∂D, so we have that

$$\begin{aligned}
\kappa^F_{\gamma_0} &= \sum_{f^* \in \gamma_0} \alpha_{\gamma_0}(f^*) \\
&= \sum_{f^* \in \gamma} \alpha_\gamma(f^*) + \sum_{f^* \in \delta} \alpha_\delta(f^*) + \sum_{f^* \in \partial D} \alpha_{\partial D}(f^*) - 4\pi \\
&= \kappa^F_\gamma + \kappa^F_\delta - 2\pi.
\end{aligned}$$

Thus, we have that

$$k^F_{\gamma_0} = k^F_\gamma + k^F_\delta - 1.$$

The proof where α is on the left hand side of the two loops is analogous. □

6.10 Proof of Proposition 2

Proof. We have that cutting M along $\gamma_1, \ldots, \gamma_{2g}$ results in a disk, so, if the mesh is sufficiently refined, for any γ_i there is a path α_j/α'_j (fig. 6.11 left) from either side of γ_i to any v_j^* such that neither path intersects any of the other basis loops or cones. Thus, by the above, we may reroute γ_i around v_j clockwise or counterclockwise to obtain a new loop γ'_i such that $H' = (H \setminus \{\gamma_i\}) \cup \{\gamma'_i\}$ also cuts M into a topological disk and such that, for any seamless parametrization F satisfying

the properties listed in the proposition, we have

$$k^F_{\gamma_i'} = k^F_{\gamma_i} \pm I^F_{v_j} = k^F_{\gamma_i} \pm I_j$$

Since H' still cuts M to a disk, we may still reroute any loop around any cone with either orientation, so we may iteratively reroute the basis loops to modify their holonomy number by integer multiples of I_j. Since $\frac{1}{4}$ is the greatest common divisor of $I_1, ..., I_m$, we have there are integers a_i such that

$$\frac{1}{4} = \sum_{i=1}^{m} a_i I_i$$

Thus, we have that iteratively rerouting each loop γ_i around the cone v_j $|4k_i a_j|$ times, with orientation determined by the signs of k_i and a_j, will give us the desired system of loops. □

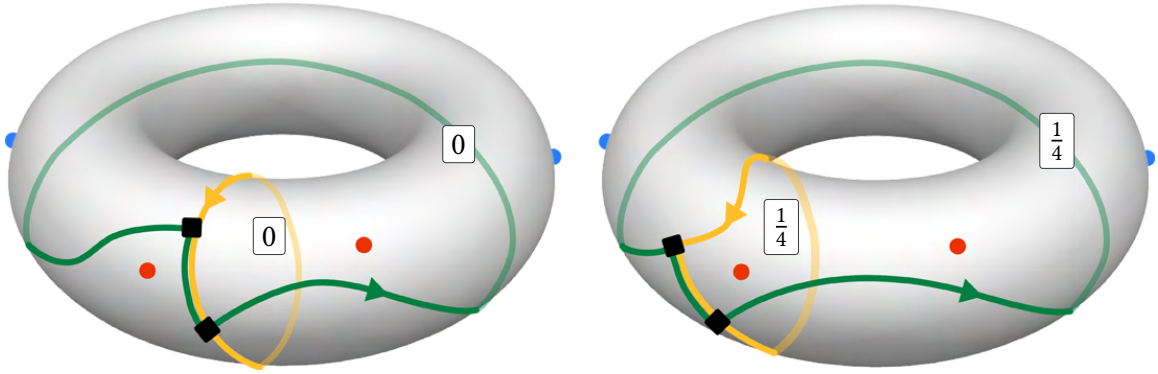


Figure 6.5: Example of two equivalent holonomy signatures. Red and blue cones have index $-\frac{1}{4}$ and $+\frac{1}{4}$, respectively; the holonomy numbers of the green and yellow loops are indicated. Note that from left to right, the loops are essentially deformed across a cone (the leftmost red cone), and this affects the loops' associated holonomy numbers accordingly.

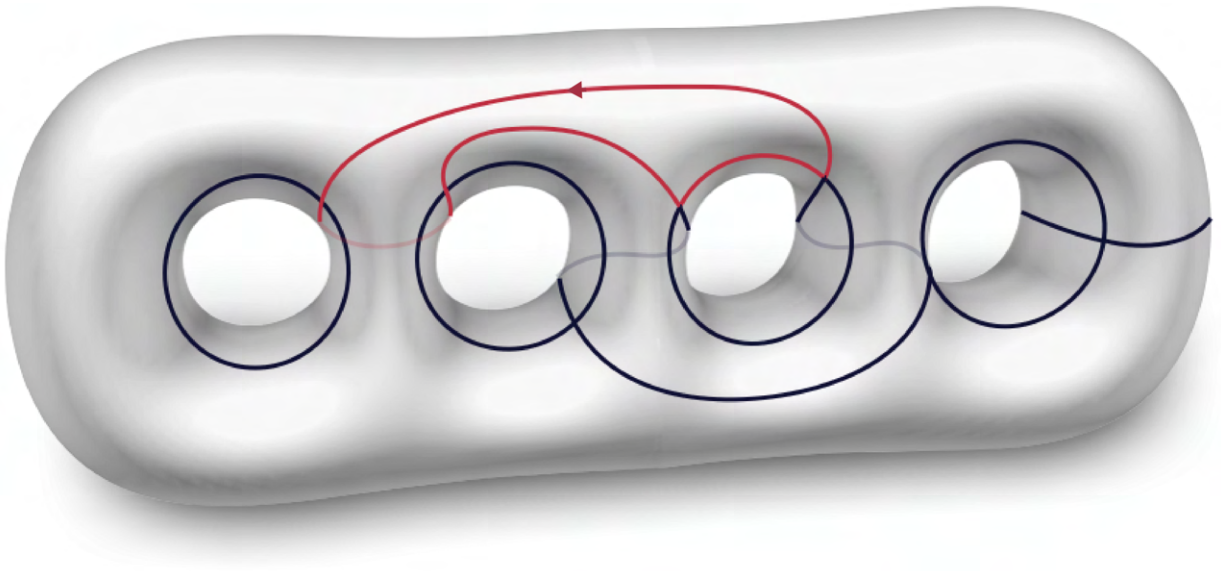


Figure 6.6: A hole-chain cut graph G, as used in [Campen et al. 2019]. As an example, the contained loop that is highlighted in red, because it makes two left turns (in ccw sense), will have holonomy number $\frac{2}{4}$ in the parametrization constructed by that method.

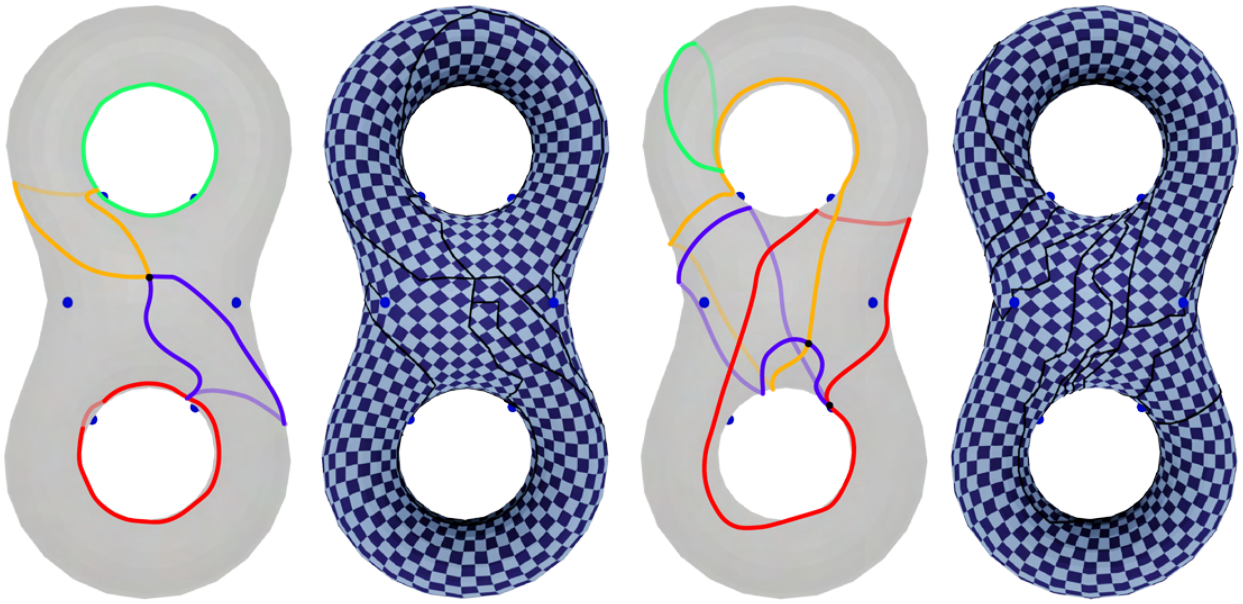


Figure 6.7: Two equivalent holonomy signatures, based on different signature loops; the different associated holonomy numbers are not shown in the figure. Both are the result of rerouting so as to achieve the required holonomy pattern, therefore the resulting optimized seamless parametrizations based on the cut graphs formed by these loop systems are identical (up to seamless transformation, due to a differently located cut graph).

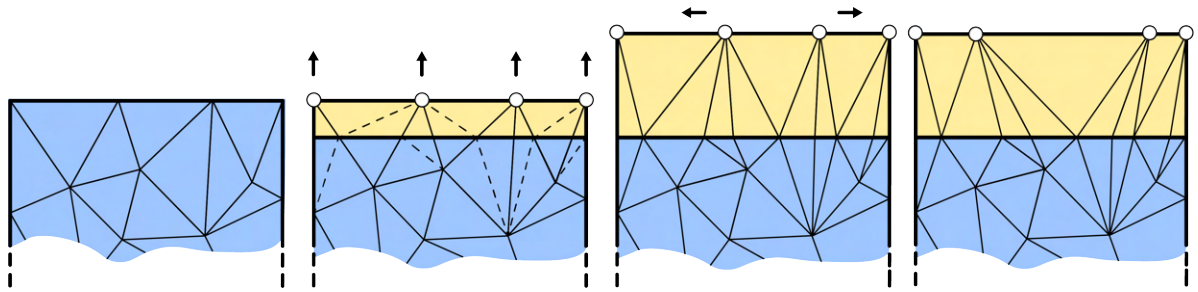


Figure 6.8: Illustration of padding operation (in parameter domain). A thin strip along the top straight cut segment (with no interior vertices) is stretched in vertical direction by its required padding width. Then, vertices are shifted horizontally to match their mates across the cut.

Table 6.1: Statistics about the number of cut segment reroutings performed. It is further split into the numbers of field-guided and fallback reroutings.

Model	Genus	#Reroutings	#Field-Guided	#Fallback
twirl	1	0	0	0
robocatdeci	1	0	0	0
knot1	1	0	0	0
holes3	3	0	0	0
dancer2	1	0	0	0
sculpt	2	0	0	0
fertilitytri	4	0	0	0
rockerarm	1	1	1	0
genus3	3	1	1	0
elk	1	1	1	0
trimstar	1	1	1	0
wrench50K	1	1	1	0
bumpytorus	1	1	1	0
dancer25k	1	1	1	0
camel	1	1	1	0
dragonstandrecon	1	1	1	0
pulley	1	1	1	0
kitten	1	1	1	0
knot	1	1	1	0
mastercylinder	3	1	1	0
eight	2	1	1	0
femur	2	2	2	0
block	3	2	2	0
greeksculpture	4	2	2	0
elephant	3	2	2	0
thaistatue	3	2	2	0
oilpump	4	2	2	0
neptune0	3	2	2	0
carter	7	2	2	0
cup	2	2	2	0
botijo	5	3	3	0
chair	7	3	3	0
rollingstage	7	3	3	0
helmet	3	4	2	2
pegaso	6	4	4	0
chair	7	4	4	0
bozbezbozzel	5	5	5	0
dancingchildren	8	5	5	0
grayloc	9	6	6	0
seahorse2	8	10	5	5
raptor50K	10	12	6	6
heptoroid	22	15	14	1
gearbox	78	57	43	14
filigree	65	73	40	33
brain	57	83	70	13
vhskin	79	128	21	107

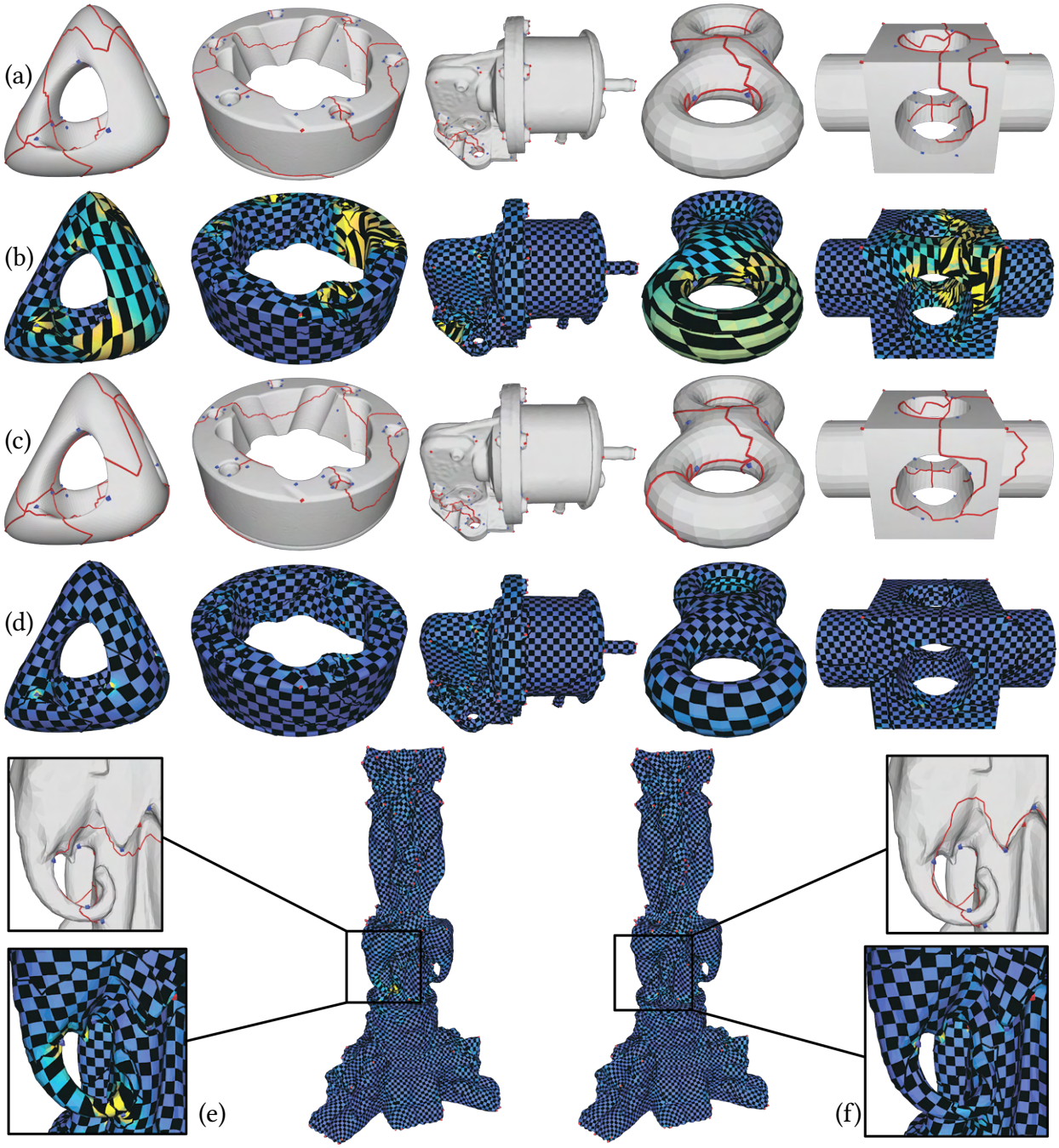


Figure 6.9: Comparison of seamless parametrizations on surfaces of non-trivial topology, computed by the bare SP method [Campen et al. 2019] (row b, e) and by our method (row d, f). The used cut graphs are shown in red, the initial hole-chain used for SP (row a, e) and the rerouted version used by our method (row c, f). Notice their topologically differing structure (i.e. they wind around some handles or cones differently), as well as the higher distortion of the results by the bare SP method due to being unable to properly align to the underlying smooth cross-field for topological reasons. Notice that this distortion cannot be reduced further by continuous optimization; there are topological obstacles.

Table 6.2: Residual energy (normalized by surface area) for the models from fig. 6.9. The columns "without rerouting" correspond to the direct application of SP, without regard for global holonomy. From the last column the advantage in terms of field alignment and distortion becomes clear.

		with rerouting		without rerouting	
Model	E_A+sE_D	E_A	E_A+sE_D	E_A	**ours/SP**
block	0.0136	0.0136	0.1115	0.1052	12.2%
eight	0.0350	0.0328	0.1524	0.1432	22.9%
genus3	0.0221	0.0208	0.1891	0.1747	11.7%
oilpump	0.0296	0.0293	0.0450	0.0370	65.7%
rollingstage	0.0127	0.0124	0.2666	0.1163	4.8%
thaistatue	0.0225	0.0225	0.0272	0.0257	82.7%

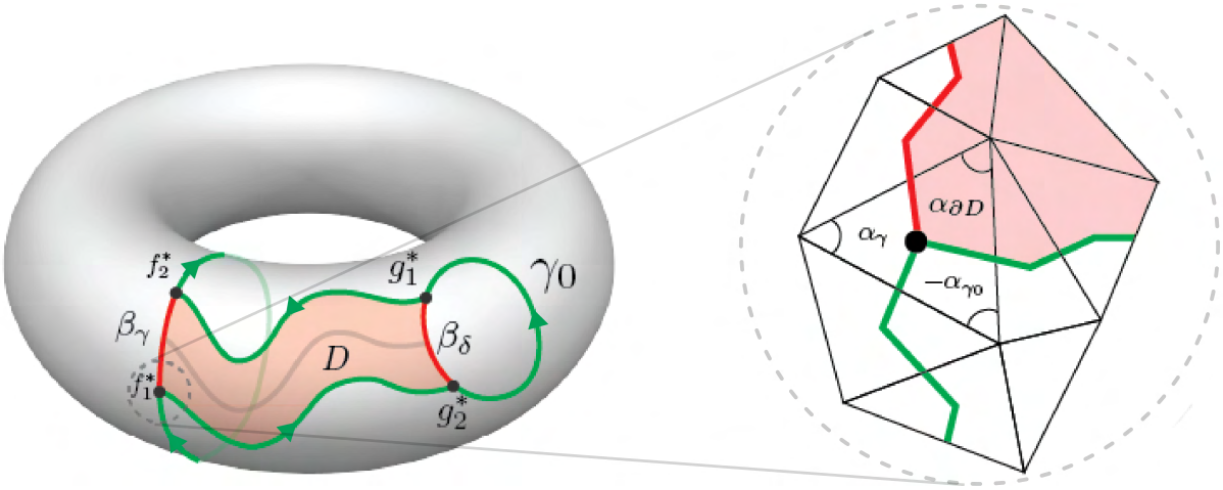


Figure 6.10: Quasi-additivity of holonomy numbers, on the same example as in fig. 6.1. The inset on the right is a blow-up of the spot circled on the left.

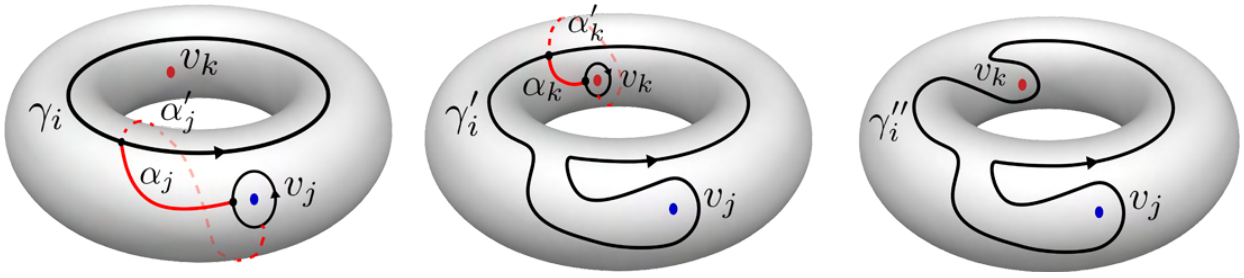


Figure 6.11: Example of iteratively rerouting one loop around two singularities. Left: initial state with given loop γ_i and two paths α_j, α'_j connecting to the singularity v_j. Center: reroute around singularity v_j and find paths α_k α'_k for the next singularity v_k. Right: result after rerouting around v_j and v_k

7 | Conclusion

In this dissertation, we investigate different types of user-prescribed constraints for the need of different applications, and present robust algorithms for generating surface parameterization with guarantees over these prescribed constraints.

In the first part, we introduced a robust algorithm, Progressive Embedding, to compute planar embeddings more reliably than Tutte's Embedding in practice. And we demonstrated its practical utility in common geometry processing tasks. Combining our progressive embedding algorithm and the matchmaker algorithm, we describe an algorithm for generating parameterization bitwise exactly satisfies the user-defined positional constraints.

In the second part, we presented a practical realization of the method for computing discrete conformal maps based on the ideas of [Gu et al. 2018b; Springborn 2020], elaborating how it can be applied safely to meshes with boundary, the most practically relevant scenario for conformal mapping. Our improvements include a straightforward to implement algorithm for maintaining symmetric Delaunay triangulations and several improvements increasing the robustness of Newton's optimization method in the context of our application. We explored its behavior on a standard dataset and for a number of challenging synthetic examples, demonstrating its robustness for a broad range of cases involving high distortion. We also observe that common failure cases can be addressed by using extended precision arithmetic, albeit at a significant cost in run time.

In the third part, we provide a general path to obtaining seamless parameterization with a

given set of cones. On a conceptual level, the approach is simple: pad a parameterization that maps cut segments to straight lines, with padding determined by solving a linear system. Our algorithm demonstrates that for (almost) any user-specified or automatically computed choice of cones, a corresponding global parameterization can be constructed, without introducing additional cones.

Lastly, we explored the relation between cross-fields and seamless surface parameterizations (and therefore quadrangulations) on a topological level. A key insight is that there are hardly any practically important obstacles to generating a seamless parameterization (or quadrangulation) that topologically matches a given cross-field. We have described a method to generate such a seamless parameterization, given an input cross-field or an abstract topological specification in form of a holonomy signature.

References

Aigerman, N., Kovalsky, S. Z., and Lipman, Y. (2017). Spherical orbifold tutte embeddings. *ACM Trans. Graph.*, 36(4):90:1–90:13.

Aigerman, N. and Lipman, Y. (2015a). Orbifold tutte embeddings. *ACM Trans. Graph.*, 34(6):190:1–190:12.

Aigerman, N. and Lipman, Y. (2015b). Orbifold tutte embeddings. *ACM Trans. Graph.*, 34(6):190:1–190:12.

Aigerman, N. and Lipman, Y. (2016a). Hyperbolic orbifold tutte embeddings. *ACM Trans. Graph.*, 35(6):217:1–217:14.

Aigerman, N. and Lipman, Y. (2016b). Hyperbolic orbifold tutte embeddings. *ACM Trans. Graph.*, 35(6):217:1–217:14.

Aigerman, N., Poranne, R., and Lipman, Y. (2014). Lifted bijections for low distortion surface mappings. *ACM Trans. Graph.*, 33(4):69:1–69:12.

Basch, J., Guibas, L. J., and Hershberger, J. (1999). Data structures for mobile data. *Journal of Algorithms*, 31(1):1–28.

Ben-Chen, M., Gotsman, C., and Bunin, G. (2008). Conformal Flattening by Curvature Prescription and Metric Scaling. *Computer Graphics Forum*.

Bobenko, A. I. and Springborn, B. A. (2007). A discrete laplace–beltrami operator for simplicial surfaces. *Discrete & Computational Geometry*, 38(4):740–756.

Bommes, D., Campen, M., Ebke, H.-C., Alliez, P., and Kobbelt, L. (2013a). Integer-grid maps for reliable quad meshing. *ACM Trans. Graph.*, 32(4):98:1–98:12.

Bommes, D., Campen, M., Ebke, H.-C., Alliez, P., and Kobbelt, L. (2013b). Integer-grid maps for reliable quad meshing. *ACM Trans. Graph.*, 32(4):98:1–98:12.

Bommes, D., Lévy, B., Pietroni, N., Puppo, E., a, C. S., Tarini, M., and Zorin, D. (2012). State of the art in quad meshing. In *Eurographics STARS*.

Bommes, D., Lévy, B., Pietroni, N., Puppo, E., Silva, C., Tarini, M., and Zorin, D. (2013c). Quad-mesh generation and processing: A survey. In *Computer Graphics Forum*. Wiley Online Library.

Bommes, D., Zimmer, H., and Kobbelt, L. (2009a). Mixed-integer quadrangulation. *ACM Trans. Graph.*, 28(3):77:1–77:10.

Bommes, D., Zimmer, H., and Kobbelt, L. (2009b). Mixed-integer quadrangulation. *ACM Trans. Graph.*, 28(3):77.

Bright, A., Chien, E., and Weber, O. (2017). Harmonic global parametrization with rational holonomy. *ACM Trans. Graph.*, 36(4).

Brönnimann, H., Fabri, A., Giezeman, G.-J., Hert, S., Hoffmann, M., Kettner, L., Pion, S., and Schirra, S. (2018). 2D and 3D linear geometry kernel. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.13 edition.

Bunin, G. (2008). A continuum theory for unstructured mesh generation in two dimensions. *Comput. Aided Geom. Des.*, 25(1):14–40.

Campen, M., Bommes, D., and Kobbelt, L. (2012). Dual loops meshing: Quality quad layouts on manifolds. *ACM Trans. Graph.*, 31(4).

Campen, M., Bommes, D., and Kobbelt, L. (2015). Quantized global parametrization. *ACM Trans. Graph.*, 34(6):192.

Campen, M., Capouellez, R., Shen, H., Zhu, L., Panozzo, D., and Zorin, D. (2021). Efficient and robust discrete conformal equivalence with boundary. *ACM Trans. Graph.*, 40(6).

Campen, M. and Kobbelt, L. (2014). Dual strip weaving: Interactive design of quad layouts using elastica strips. *ACM Trans. Graph.*, 33(6):183:1–183:10.

Campen, M., Shen, H., Zhou, J., and Zorin, D. (2019). Seamless parametrization with arbitrary cones for arbitrary genus. *ACM Trans. Graph.*, 39(1).

Campen, M., Silva, C. T., and Zorin, D. (2016). Bijective maps from simplicial foliations. *ACM Trans. Graph.*, 35(4):74:1–74:15.

Campen, M. and Zorin, D. (2017a). On discrete conformal seamless similarity maps.

Campen, M. and Zorin, D. (2017b). Similarity maps and field-guided t-splines: A perfect couple. *ACM Trans. Graph.*, 36(4).

Chambers, E. W., Eppstein, D., Goodrich, M. T., and Löffler, M. (2011). Drawing graphs in the plane with a prescribed outer face and polynomial area. In *Proceedings of the 18th International Conference on Graph Drawing*, GD'10, pages 129–140, Berlin, Heidelberg. Springer-Verlag.

Chen, W., Zheng, X., Ke, J., Lei, N., Luo, Z., and Gu, X. (2019). Quadrilateral mesh generation i: Metric based method. *Computer Methods in Applied Mechanics and Engineering*, 356:652–668.

Chen, W., Zheng, X., Ke, J., Lei, N., Luo, Z., and Gu, X. (2020). Quadrilateral mesh generation ii: Meoromorphic quartic differentials and abel-jacobi condition. *Computer Methods in Applied Mechanics and Engineering*, 366.

Cherrier, P. (1984). Problèmes de neumann non linéaires sur les variétés riemanniennes. *Journal of Functional Analysis*, 57(2):154–206.

Chien, E., Levi, Z., and Weber, O. (2016). Bounded distortion parametrization in the space of metrics. *ACM Trans. Graph.*, 35(6).

Claici, S., Bessmeltsev, M., Schaefer, S., and Solomon, J. (2017). Isometry-aware preconditioning for mesh parameterization. *Comput. Graph. Forum*, 36(5):37–47.

Crane, K. (2020). Discrete Conformal Geometry. In *Proceedings of Symposia in Applied Mathematics*. American Mathematical Society.

Crane, K., Desbrun, M., and Schröder, P. (2010). Trivial connections on discrete surfaces. *Computer Graphics Forum*, 29(5):1525–1533.

Davis, P. J. (2012).

Degener, P., Meseth, J., and Klein, R. (2003). An adaptable surface parameterization method. In *Proceedings of the 12th International Meshing Roundtable*, pages 201–213.

Desbrun, M., Meyer, M., and Alliez, P. (2002). Intrinsic parameterizations of surface meshes. In *Computer graphics forum*, pages 209–218. Wiley Online Library.

Dey, T. K., Edelsbrunner, H., Guha, S., and Nekhayev, D. V. (1999). Topology preserving edge contraction. *Publ. Inst. Math.(Beograd)(NS)*, 66(80):23–45.

Dey, T. K., Fan, F., and Wang, Y. (2013). An efficient computation of handle and tunnel loops via reeb graphs. *ACM Trans. Graph.*, 32(4).

Diaz-Gutierrez, P., Eppstein, D., and Gopi, M. (2009). Curvature aware fundamental cycles. In *Computer Graphics Forum*, pages 2015–2024. Wiley Online Library.

Ebke, H.-C., Bommes, D., Campen, M., and Kobbelt, L. (2013). QEx: Robust quad mesh extraction. *ACM Trans. Graph.*, 32(6):168:1–168:10.

Ebke, H.-C., Schmidt, P., Campen, M., and Kobbelt, L. (2016). Interactively controlled quad remeshing of high resolution 3d models. *ACM Trans. Graph.*, 35(6):218:1–218:13.

Erickson, J. and Whittlesey, K. (2005). Greedy optimal homotopy and homology generators. In *SODA*, volume 5, pages 1038–1046.

Fáry, I. (1948). On straight line representation of planar graphs. *Acta Univ. Szeged. Sect. Sci. Math.*, 11:229–233.

Fisher, M., Springborn, B., Schröder, P., and Bobenko, A. I. (2007). An algorithm for the construction of intrinsic delaunay triangulations with applications to digital geometry processing. *Computing*, 81(2-3):199–213.

Floater, M. and Hormann, K. (2005a). Surface Parameterization: a Tutorial and Survey. *Advances In Multiresolution For Geometric Modelling*.

Floater, M. S. (1997a). Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14:231–250.

Floater, M. S. (1997b). Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14(3):231 – 250.

Floater, M. S. and Hormann, K. (2005b). Surface parameterization: a tutorial and survey. In *In Advances in Multiresolution for Geometric Modelling, Mathematics and Visualization*, pages 157–186. Springer Verlag.

Fousse, L., Hanrot, G., Lefèvre, V., Pélissier, P., and Zimmermann, P. (2007). Mpfr: A multiple-precision binary floating-point library with correct rounding. *ACM Trans. Math. Softw.*, 33(2).

Fu, X.-M. and Liu, Y. (2016). Computing inversion-free mappings by simplex assembly. *ACM Trans. Graph.*, 35(6):216:1–216:12.

Fu, X.-M., Liu, Y., and Guo, B. (2015a). Computing locally injective mappings by advanced mips. *ACM Trans. Graph.*, 34(4):71:1–71:12.

Fu, X.-M., Liu, Y., and Guo, B. (2015b). Computing locally injective mappings by advanced mips. *ACM Trans. Graph.*, 34(4).

Gillespie, M., Springborn, B., and Crane, K. (2021). Discrete conformal equivalence of polyhedral surfaces. *ACM Trans. Graph.*, 40(4).

Gortler, S. J., Gotsman, C., and Thurston, D. (2006). Discrete one-forms on meshes and applications to 3d mesh parameterization. *Computer Aided Geometric Design*, 23(2):83 – 112.

Gotsman, C. and Surazhsky, V. (2001). Guaranteed intersection-free polygon morphing. *Computers & Graphics*, 25(1):67–75.

Granlund, T. (2018). *GNU MP: The GNU Multiple Precision Arithmetic Library*, 5.0.5 edition. http://gmplib.org/.

Grisvard, P. (1985). Elliptic problems in nonsmooth domains, volume 24 of monographs and studies in mathematics. pitman.

Gu, X., Guo, R., Luo, F., Sun, J., and Wu, T. (2014). A discrete uniformization theorem for polyhedral surfaces ii.

Gu, X., Guo, R., Luo, F., Sun, J., and Wu, T. (2018a). A discrete uniformization theorem for polyhedral surfaces ii. *Journal of differential geometry*, 109(3):431–466.

Gu, X., Luo, F., Sun, J., and Wu, T. (2013). A discrete uniformization theorem for polyhedral surfaces.

Gu, X. and Yau, S.-T. (2003). Global conformal surface parameterization. In *Proc. Symp. Geometry Processing 2003*, pages 127–137.

Gu, X. D., Luo, F., Sun, J., and Wu, T. (2018b). A discrete uniformization theorem for polyhedral surfaces. *Journal of differential geometry*, 109(2):223–256.

Guennebaud, G., Jacob, B., et al. (2010). Eigen v3. http://eigen.tuxfamily.org.

Halperin, D. and Packer, E. (2002). Iterated snap rounding. *Computational Geometry*, 23(2):209 – 225.

Hatcher, A. (2002). *Algebraic Topology*. Cambridge University Press.

Hefetz, E. F., Chien, E., and Weber, O. (2019). A subspace method for fast locally injective harmonic mapping. In *Computer Graphics Forum*, pages 105–119.

Hoppe, H. (1996). Progressive meshes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 99–108, New York, NY, USA. ACM.

Hormann, K. and Greiner, G. (2012). Mips: An efficient global parametrization method. *Curve and Surface Design: Saint-Malo*, 2000:10.

Hormann, K., Lévy, B., and Sheffer, A. (2007). Mesh parameterization: Theory and practice. In *ACM SIGGRAPH 2007 Courses*, SIGGRAPH '07, New York, NY, USA. ACM.

Hu, Y., Zhou, Q., Gao, X., Jacobson, A., Zorin, D., and Panozzo, D. (2018). Tetrahedral meshing in the wild. *ACM Trans. Graph.*, 37(4):60:1–60:14.

Hudson, J. F. and Shaneson, J. L. (1969). *Piecewise linear topology*, volume 11. WA Benjamin New York.

Jacobson, A., Panozzo, D., et al. (2016). libigl: A simple C++ geometry processing library. http://libigl.github.io/libigl/.

Jiang, Z., Schaefer, S., and Panozzo, D. (2017). Simplicial complex augmentation framework for bijective maps. *ACM Trans. Graph.*, 36(6):186:1–186:9.

Jin, M., Kim, J., and Gu, X. D. (2007). Discrete surface ricci flow: Theory and applications. In *IMA International Conference on Mathematics of Surfaces*, pages 209–232. Springer.

Jucovič, E. and Trenkler, M. (1973). A theorem on the structure of cell–decompositions of orientable 2–manifolds. *Mathematika*, 20(01):63–82.

Kälberer, F., Nieser, M., and Polthier, K. (2007). QuadCover: Surface Parameterization using Branched Coverings. *Computer Graphics Forum*, 26(3):375–384.

Kharevych, L., Springborn, B., and Schröder, P. (2005). Discrete conformal mappings via circle patterns. In *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05, page 6–es, New York, NY, USA. Association for Computing Machinery.

Kharevych, L., Springborn, B., and Schröder, P. (2006a). Discrete conformal mappings via circle patterns. *ACM Transactions on Graphics (TOG)*, 25(2):412–438.

Kharevych, L., Springborn, B., and Schröder, P. (2006b). Discrete conformal mappings via circle patterns. *ACM Trans. Graph.*, 25:412–438.

Knupp, P. (1995). Mesh generation using vector fields. *Journal of Computational Physics*, 119(1):142 – 148.

Kovalsky, S. Z., Aigerman, N., Basri, R., and Lipman, Y. (2015). Large-scale bounded distortion mappings. *ACM Trans. Graph.*, 34(6):191:1–191:10.

Kovalsky, S. Z., Galun, M., and Lipman, Y. (2016a). Accelerated quadratic proxy for geometric optimization. *ACM Trans. Graph.*, 35(4):134:1–134:11.

Kovalsky, S. Z., Galun, M., and Lipman, Y. (2016b). Accelerated quadratic proxy for geometric optimization. *ACM Trans. Graph.*, 35(4):134:1–134:11.

Kraevoy, V. and Sheffer, A. (2004). Cross-parameterization and compatible remeshing of 3d models. *ACM Trans. Graph.*, 23(3):861–869.

Kraevoy, V., Sheffer, A., and Gotsman, C. (2003). Matchmaker: Constructing constrained texture maps. *ACM Trans. Graph.*, 22(3):326–333.

Kucera, L. (1991). The greedy coloring is a bad probabilistic algorithm. *Journal of Algorithms*, 12(4):674 – 684.

Labsik, U., Hormann, K., and Greiner, G. (2000). Using most isometric parametrizations for remeshing polygonal surfaces. In *Proceedings of the Geometric Modeling and Processing 2000*, GMP 2000, Washington, DC, USA. IEEE Computer Society.

Lee, T. Y., Yen, S. W., and Yeh, I. C. (2008). Texture mapping with hard constraints using warping scheme. *IEEE Transactions on Visualization and Computer Graphics*, 14(2):382–395.

Lévy, B., Petitjean, S., Ray, N., and Maillot, J. (2002a). Least squares conformal maps for automatic texture atlas generation. *ACM transactions on graphics (TOG)*, 21(3):362–371.

Lévy, B., Petitjean, S., Ray, N., and Maillot, J. (2002b). Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph.*, 21(3):362–371.

Li, M., Kaufman, D. M., Kim, V. G., Solomon, J., and Sheffer, A. (2018). Optcuts: joint optimization of surface cuts and parameterization. In *SIGGRAPH Asia 2018 Technical Papers*, page 247. ACM.

Li, W., Vallet, B., Ray, N., and Levy, B. (2006). Representing higher-order singularities in vector fields on piecewise linear surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1315–1322.

Lipman, Y. (2012). Bounded distortion mapping spaces for triangular meshes. *ACM Trans. Graph.*, 31(4):108:1–108:13.

Lipman, Y. (2014). Bijective mappings of meshes with boundary and the degree in mesh processing. *SIAM Journal on Imaging Sciences*, 7(2):1263–1283.

Liu, L., Ye, C., Ni, R., and Fu, X.-M. (2018). Progressive parameterizations. *ACM Trans. Graph.*, 37(4):41:1–41:12.

Luo, F. (2004). Combinatorial yamabe flow on surfaces. *Communications in Contemporary Mathematics*, 6(05):765–780.

Lyon, M., Campen, M., Bommes, D., and Kobbelt, L. (2019). Parametrization quantization with free boundaries for trimmed quad meshing. *ACM Trans. Graph.*, 38(4).

Mandad, M. and Campen, M. (2019). Exact constraint satisfaction for truly seamless parametrization. *Computer Graphics Forum*, 38.

Mijatović, A. (2003). Simplifying triangulations of s^3. *Pacific journal of mathematics*, 208(2):291–324.

Müller, M., Chentanez, N., Kim, T.-Y., and Macklin, M. (2015). Air meshes for robust collision handling. *ACM Trans. Graph.*, 34(4):133:1–133:9.

Myles, A., Pietroni, N., and Zorin, D. (2014). Robust field-aligned global parametrization. *ACM Trans. Graph.*, 33(4):135:1–135:14.

Myles, A. and Zorin, D. (2012). Global parametrization by incremental flattening. *ACM Trans. Graph.*, 31(4):109.

Myles, A. and Zorin, D. (2013). Controlled-distortion constrained global parametrization. *ACM Transactions on Graphics*, 32(4):105.

Packer, E. (2018). 2D snap rounding. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.13 edition.

Panozzo, D., Lipman, Y., Puppo, E., and Zorin, D. (2012). Fields on symmetric surfaces. *ACM Trans. Graph.*, 31(4).

Peng, Y., Deng, B., Zhang, J., Geng, F., Qin, W., and Liu, L. (2018). Anderson acceleration for geometry optimization and physics simulation. *ACM Trans. Graph.*, 37(4):42:1–42:14.

Poranne, R. and Lipman, Y. (2014). Provably good planar mappings. *ACM Trans. Graph.*, 33(4):76:1–76:11.

Poranne, R., Tarini, M., Huber, S., Panozzo, D., and Sorkine-Hornung, O. (2017). Autocuts: Simultaneous distortion and cut optimization for uv mapping. *ACM Trans. Graph.*, 36(6):215:1–215:11.

Praun, E., Sweldens, W., and Schröder, P. (2001). Consistent mesh parameterizations. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 179–184, New York, NY, USA. ACM.

Purnomo, B., Cohen, J. D., and Kumar, S. (2004). Seamless Texture Atlases. In Scopigno, R. and Zorin, D., editors, *Symposium on Geometry Processing*. The Eurographics Association.

Rabinovich, M., Poranne, R., Panozzo, D., and Sorkine-Hornung, O. (2017a). Scalable locally injective mappings. *ACM Trans. Graph.*, 36(2):16:1–16:16.

Rabinovich, M., Poranne, R., Panozzo, D., and Sorkine-Hornung, O. (2017b). Scalable locally injective mappings. *ACM Trans. Graph.*, 36(2):16:1–16:16.

Ray, N., Nivoliers, V., Lefebvre, S., and Lévy, B. (2010). Invisible seams. *Computer Graphics Forum*, 29.

Ray, N., Vallet, B., Alonso, L., and Levy, B. (2009). Geometry-aware direction field processing. *ACM Trans. Graph.*, 29(1).

Ray, N., Vallet, B., Li, W. C., and Lévy, B. (2008). N-symmetry direction field design. *ACM Trans. Graph.*, 27(2).

Rivin, I. (1994). Euclidean structures on simplicial surfaces and hyperbolic volume. *Annals of mathematics*, 139(3):553–580.

Sander, P. V., Snyder, J., Gortler, S. J., and Hoppe, H. (2001). Texture mapping progressive meshes. In *ACM SIGGRAPH*, pages 409–416.

Sawhney, R. and Crane, K. (2017). Boundary first flattening. *ACM Trans. Graph.*, 37(1).

Schmidt, P., Campen, M., Born, J., and Kobbelt, L. (2020). Inter-surface maps via constant-curvature metrics. *ACM Transactions on Graphics*, 39(4).

Schnyder, W. (1990). Embedding planar graphs on the grid. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '90, pages 138–148, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Schreiner, J., Asirvatham, A., Praun, E., and Hoppe, H. (2004). Inter-surface mapping. *ACM Trans. Graph.*, 23(3):870–877.

Schüller, C., Kavan, L., Panozzo, D., and Sorkine-Hornung, O. (2013). Locally injective mappings. In *Proceedings of the Eleventh Eurographics/ACMSIGGRAPH Symposium on Geometry Processing*, SGP '13, pages 125–135, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

Schüller, C., Kavan, L., Panozzo, D., and Sorkine-Hornung, O. (2013). Locally injective mappings. *Computer Graphics Forum*, 32(5):125–135.

Segall, A., Vantzos, O., and Ben-Chen, M. (2016). Hele-shaw flow simulation with interactive control using complex barycentric coordinates. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation*, pages 85–95. Eurographics Association.

Sharp, N. and Crane, K. (2020). A laplacian for nonmanifold triangle meshes. In *Computer Graphics Forum*, pages 69–80. Wiley Online Library.

Sharp, N., Soliman, Y., and Crane, K. (2019). Navigating intrinsic triangulations. *ACM Transactions on Graphics (TOG)*, 38(4):1–16.

Sheffer, A., Praun, E., and Rose, K. (2006). Mesh parameterization methods and their applications. *Found. Trends. Comput. Graph. Vis.*, 2(2):105–171.

Shewchuk, J. R. (1996). Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In Lin, M. C. and Manocha, D., editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag. From the First ACM Workshop on Applied Computational Geometry.

Shor, P. W. and Van Wyk, C. J. (1992). Detecting and decomposing self-overlapping curves. *Computational Geometry*, 2(1):31–50.

Shtengel, A., Poranne, R., Sorkine-Hornung, O., Kovalsky, S. Z., and Lipman, Y. (2017a). Geometric optimization via composite majorization. *ACM Trans. Graph.*, 36(4):38:1–38:11.

Shtengel, A., Poranne, R., Sorkine-Hornung, O., Kovalsky, S. Z., and Lipman, Y. (2017b). Geometric optimization via composite majorization. *ACM Trans. Graph.*, 36(4):38:1–38:11.

Smith, J. and Schaefer, S. (2015). Bijective parameterization with free boundaries. *ACM Trans. Graph.*, 34(4).

Soliman, Y., Slepčev, D., and Crane, K. (2018a). Optimal cone singularities for conformal flattening. *ACM Transactions on Graphics (TOG)*, 37(4):1–17.

Soliman, Y., Slepčev, D., and Crane, K. (2018b). Optimal cone singularities for conformal flattening. *ACM Trans. Graph.*, 37(4):105:1–105:17.

Sorkine, O., Cohen-Or, D., Goldenthal, R., and Lischinski, D. (2002). Bounded-distortion piecewise mesh parameterization. In *Proceedings of the Conference on Visualization*, pages 355–362.

Springborn, B. (2017). Hyperbolic polyhedra and discrete uniformization.

Springborn, B. (2019). Ideal hyperbolic polyhedra and discrete uniformization. *Discrete & Computational Geometry*, pages 1–46.

Springborn, B., Schröder, P., and Pinkall, U. (2008). Conformal equivalence of triangle meshes. *ACM Trans. Graph.*, 27(3):1–11.

Sun, J., Wu, T., Gu, X., and Luo, F. (2015). Discrete conformal deformation: algorithm and experiments. *SIAM Journal on Imaging Sciences*, 8(3):1421–1456.

Tong, Y., Alliez, P., Cohen-Steiner, D., and Desbrun, M. (2006). Designing quadrangulations with discrete harmonic forms. *Symposium on Geometry Processing*, pages 201–210.

Troyanov, M. (1991). Prescribing curvature on compact surfaces with conical singularities. *Transactions of the American Mathematical Society*, 324(2):793–821.

Tutte, W. (1963a). How to draw a graph. *Proc. London Math. Soc.,*, 3:743–768.

Tutte, W. T. (1963b). How to draw a graph. *Proc. Lond. Math. Soc.*, 13:743–767.

Vaxman, A., Campen, M., Diamanti, O., Panozzo, D., Bommes, D., Hildebrandt, K., and Ben-Chen, M. (2016). Directional field synthesis, design, and processing. *Comp. Graph. Forum*, 35(2).

Weber, O. and Zorin, D. (2014). Locally injective parametrization with arbitrary fixed boundaries. *ACM Trans. Graph.*, 33(4):75:1–75:12.

Weeks, J. R. (1993). Convex hulls and isometries of cusped hyperbolic 3-manifolds. *Topology and its Applications*, 52(2):127–149.

Wu, T. (2014). Finiteness of switches in discrete yamabe flow. Master's thesis, Tsinghua University.

Zhang, E., Mischaikow, K., and Turk, G. (2005). Feature-based surface parameterization and texture mapping. *ACM Trans. Graph.*, 24(1):1–27.

Zhou, J., Tu, C., Zorin, D., and Campen, M. (2020). Combinatorial construction of seamless parameter domains. *Computer Graphics Forum*, 39(2):179–190.

Zhou, Q. and Jacobson, A. (2016). Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797*.

Zhu, Y., Bridson, R., and Kaufman, D. M. (2018a). Blended cured quasi-newton for distortion optimization. *ACM Trans. Graph.*, 37(4):40:1–40:14.

Zhu, Y., Bridson, R., and Kaufman, D. M. (2018b). Blended cured quasi-newton for distortion optimization. *ACM Trans. Graph.*, 37(4):40:1–40:14.

Zorin, D. (2021). Convergence analysis of the algorithm in "efficient and robust discrete conformal equivalence with boundary".